Improvement of Decoding Engine & Phonetic Decision Tree in Acoustic Modeling for Online Large Vocabulary Conversational Speech Recognition

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of requirements for the Degree

Doctor of Philosophy

by

Jian Xue

Dr. Yunxin Zhao, Dissertation Supervisor

December 2007

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

IMPROVEMENT OF DECODING ENGINES AND PHONETIC DECISION TREES IN ACOUSTIC MODELING FOR ONLINE LARGE VOCABULARY CONVERSATIONAL SPEECH RECOGNITION

presented by Jian Xue, a candidate for the degree of doctor of philosophy, and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Yunxin Zhao

Professor Xinhua Zhuang

Professor Hongchi Shi

Professor Dong Xu

Professor Dominic Ho

Acknowledgements

I will begin by thanking my advisor, Dr. Yunxin Zhao, who guided me since I was a novice in ASR field. I always feel so lucky that I have such a great advisor in my Ph.D study. Without her knowledge, perceptiveness and guidance, I would never have finished my study.

Also, I would like to thank Dr. Xinhua Zhuang, Dr. Dong Xu, Dr. hongchi Shi, and Dr. Dominic K.C. Ho, for their help in my graduate study as my committee members. Thank them for reviewing my dissertation and giving inspirational comments.

Thank all the officemates in the Laboratory of Spoken Language and Information Processing, especially Xiaodong He, Chuanhong Ma, Xiaolong Li, Rusheng Hu, Rong Hu, Xiaojia Zhang, Yi Zhang, and Xin Chen, for their great help both in study and daily lift. Working with them has been a great experience in my whole life.

Many thanks to my wonderful wife, Lili Che, for her constant encouragement and support, and most of all for her endless love. Also thanks my mother and sister, their love are always my propulsion.

TABLE OF CONTENTS

List of Figures

List of Tables

Abstract

In this dissertation work, new approaches are proposed for online large vocabulary conversational speech recognition, including a fast confusion network algorithm for aligning competing word hypotheses, novel features and a Random Forests based classifier for word confidence annotation, new improvements in speech decoding speed and latency, novel lookahead phonetic decision tree state tying and Random Forests of phonetic decision tree state tying for acoustic modeling of speech sound units.

The fast confusion network algorithm significantly improves the time complexity from $O(T^3)$ to $O(T)$, with T equaling the number of links in a word lattice, making it feasible to use confusion network in online automatic speech recognition system. Several novel features, as well as Random Forests based classification technique are proposed to improve word annotation accuracy for automatic captioning. In order to improve the speed of speech decoding engine, we propose to use complementary word confidence scores to prune uncompetitive search paths, and use subspace distribution clustering hidden Markov modeling to speed up computation of acoustic scores and local confidence scores. We further integrate pre-backtrace in decoding search to significantly reduce captioning latency.

Phonetic decision tree based clustering are commonly used in acoustic modeling to reduce the number of context-dependent phone models, and to model speech units in some contexts which do not occur in training data. In this work we investigate novel approaches to improve the performance of phonetic decision tree state tying, including two lookahead methods and a Random Forests method. The lookahead methods overcome the weakness of conventional greedy search methods by taking into account the node splits beyond the immediate children nodes. Constrained lookahead method finds an optimal question among n pre-selected questions for each split node to decrease effects of outliers, and it also discounts the contributions of likelihood gains by deeper decedents. Stochastic full lookahead method uses sub-tree size instead of likelihood gain as a measure for phonetic question selection, in order to produce small trees with better generalization capability and consistent with training data. The Random Forests method uses an ensemble of phonetic decision trees to

derive a single strong model for each speech unit. We investigate several methods of combining the acoustic scores from multiple models obtained from multiple phonetic decision trees in decoding search. We further propose clustering methods to compact the Random Forests generated acoustic models to speed up decoding search.

The proposed new methods have been integrated into the Telemedicine automatic captioning system developed in the Spoken Language and Information Processing Laboratory of the Computer Science Department of University of Missouri - Columbia, and they have improved the system's performance in accuracy, speed, latency as well as its functionality.

Chapter 1

Introduction

1.1 Statistical Speech Recognition

Speech is the most natural means of communication among human being. However, it has been a long-time dream for humans to use speech to communicate with a computer [1]. Automatic speech recognition (ASR) systems allow people to control a computer by speaking to it through a microphone, providing options of typing text or issuing commands to the computer. The problem of speech recognition has been actively studied since the 1950's, and a number of significant advances have been made over the half century.

The role of a speech recognizer is to map a sequence of observation vectors of speech into its underlying word sequence, i.e., to find the most probable string of words $\hat{W}$ corresponding to the acoustic observation *O*. Most of current state-of-art systems for automatic speech recognition are based on Bayesian decision theory. By applying Bayesian rule on conditional probabilities, the problem of speech recognition can be written in the following form [2]:

$$\hat{W} = \arg\max_{W} p(W \mid O) = \arg\max_{W} \frac{p(O \mid W)p(W)}{p(O)} = \arg\max_{W} p(O \mid W)p(W) \qquad (1.1)$$

where $O = o_1, o_2, \cdots, o_T$ is a sequence of acoustic feature vectors in an utterance, $W = w_1, w_2, \cdots, w_n$ is the sequences of words (with unknown length *n*) intended by the speaker which generates the acoustic feature vector sequence *O*. *p(O|W)* is the probability that the speaker produces the acoustic feature vectors sequence *O* if *W* is the intended word sequence. *p(W)* is the prior probability of word sequence *W*, which is independent of observation *O*. *p(O)* is the prior probability of observation *O*, which is independent of

word sequences and would not affect the decision, and so it is deleted in the last part of formula (1.1).

Statistical modeling for estimating the probability $p(O|W)$ is called acoustic modeling, which typically consists of two parts. The first is to describe how a word sequence can be represented by sub-word units, often known as pronunciation modeling and is specified by a pronunciation dictionary. The second is the mapping from each sub-word units to acoustic observations [3]. Algorithms used in acoustic modeling involves phonetic decision tree (PDT) and Hidden Markov Model (HMM). We will introduce acoustic modeling in section 1.3.

Statistical modeling for estimating the prior probability $p(W)$ of a given word sequence W is called language modeling, which only considers a prior probability of word sequence in a sentence. The most commonly used language model is N-gram, which will be discussed in section 1.4.

Note that in practice the language model term $p(W)$ has to be scaled to give $p(W)^{\alpha}$, say, $\alpha = 14$. This is to prevent the language model probabilities being overwhelmed by the acoustic likelihood scores, which tend to be highly correlated between acoustic feature vectors and overestimate the system's confidence in the acoustic information.

1.2 Pre-processing of Speech

Speech recognition requires effective representation of speech signals. The raw data input to ASR system is the speech waveform sampled at a certain clock rate. This data is pre-processed to generate feature vectors which retain only necessary information for the speech recognition task, referred to as feature extraction. Each feature vector is often computed per 10 ms, from an overlapped sliding window of 20 to 25 ms. Well-known feature extraction algorithms include:

1. Linear Predictive Coefficient (LPC) – a speech sample at time t is approximated as a linear combination of the past p speech samples, and the combination

coefficients are assumed constant over the speech frame [4].

2. Mel Frequency Cepstral Coefficients (MFCC) – cepstrum is first computed from warping the log energy spectrum according to the Mel frequency scale and then taking the cosine transform [5].
3. Perceptual Linear Prediction (PLP) – a variation of linear prediction coefficients taking into account of human auditory perception model [6].

The above mentioned features are all considered to be short-term locally stationary features and cannot cover the temporal dynamics in speech. It is a common practice to use first-order and second-order time-derivatives of static features to capture such information [7].

Extracted features can be further transformed to improve ASR system performance. Such transformation algorithms include principal components analysis (PCA), linear discriminant analysis (LDA or HLDA [8]), vocal tract length normalization (VTLN) and independent component analysis (ICA) [9]. The ultimate goal of speech pre-processing is to produce discriminant and robust features to close the gap between the performance of human listeners and that of ASR systems. However, there is still much work remains to be done to fulfill this task.

1.3 Statistical Acoustic Modeling and Phonetic Decision Tree

Acoustic model is used to describe the acoustic-phonetic characteristics of speech signal. Speech signal is considered as the output of a stochastic process, and the generative and distribution properties of this stochastic process are described. The core statistical modeling techniques in ASR system is HMM, which is used to model the production of speech signals and to compute the acoustic score $p(O|W)$ [1].

1.3.1 HMMs in Speech Recognition

In most commonly used HMM, the speech production mechanism is treated as a

stochastic process generating the observed speech signals. The speech production mechanism is assumed to operate in a fashion that can be characterized by a series of state transitions. These state transitions are assumed to be modeled by a Markov process in that the probability of moving to the next state depends only on the identity of the current state. As shown in Fig. 1.1, a HMM is a finite state machine which changes state once every time frame, and at each time frame t when a state j is entered, an observation vector x_t is generated from the emitting probability distribution $b_j(o_t)$. The transition from state i to state j is specified by the transition probability a_{ij}. Moreover, two special non-emitting states are usually used in a HMM. They include an entry state, which is reached before the speech vector genernation process begins, and an exit state, which is reached when the genernative process terminates. Both states are reached only once. Since they do not generate any observation, none of them has an emitting probability density.

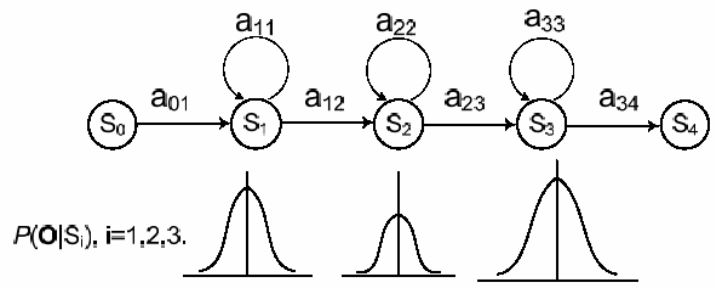


Fig. 1.1 An example of HMM for a phoneme model

In the hidden Markov model, the transition probability a_{ij} is the probability of entering state j given the previous state i, i.e.,

$$a_{ij} = P_r(s(t) = j \mid s(t-1) = i) \tag{1.2}$$

where $s(t)$ is the state index at time t. For a N-state HMM, we have $\sum_{j=1}^{N} a_{ij} = 1$ for every i.

The emitting proabability density $b_j(o)$ describes the distribution of the observation vectors at the state j. In continuous density HMM (CDHMM), emitting proabability density is often represented by a Gaussian Mixture Density (GMD)

$$b_j(o) = \sum_{m=1}^{M} c_{j,m} N(o; \mu_{jm}, \Sigma_{jm})$$

$$\sum_{m=1}^{M} c_{j,m} = 1 \quad \text{and} \quad c_{j,m} \geq 0\,, \tag{1.3}$$

where $N(o; \mu_{jm}, \Sigma_{jm}) = \frac{1}{(2\pi)^{D/2} |\Sigma_{jm}|^{1/2}} e^{-\frac{1}{2}(o-\mu_{jm})^T \Sigma_{jm}^{-1} (o-\mu_{jm})}$ is a multivariate Gaussian Density, D is the dimension of the feature vector x, and c_{jm}, μ_{jm}, and Σ_{jm} are the weight, mean and covariance of the m-th Gaussian component of the GMD at state j. Generally speaking, each emitting distribution charactizes a sound event, and the distribution must be specific enough to allow discrmination between different sounds as well as robust enough to account for the variability in natural speech.

Numerous model training methods are proposed to estimate values of the state transition probalities and the parameters of the emitting probability densties at each state. Given $\{a_{ij}\}$ and $b_j(o)$, i =1~N, j = 1~N, the likelihood of an observation sequence O given word sequence W is calculated as

$$p(O \mid W) = \sum_{S} p(O, S \mid W) \tag{1.4}$$

where $S = s_1, s_2, \ldots, s_T$ is the hidden Markov model state sequence that generates the observation vector sequence $O = o_1, o_2, \ldots, o_T$, and the joint probability of O and the state sequence S given W is a product of the transition probabilities and the emitting probabilities

$$p(O, S \mid W) = \prod_{t=1}^{T} b_{s_t}(o_t) a_{s_t s_{t+1}} \tag{1.5}$$

where s_{T+1} is the non-emitting exit state.

In practice, formula 1.4 can be approximately calculated as the joint probability of the observation vector sequence O with the most possible state sequence, i.e.,

$$p(O \mid W) \approx \max_{S} p(O, S \mid W). \tag{1.6}$$

In large vocabulary continuous speech recognition systems, it is impractical to build a HMM for each word or word sequence. In this case, sub-word units, such as syllables and phonemes, are used as the basic recognition units. A HMM is built for each sub-word unit and the model of a word string is constrcuted by concatenating the corresponding sub-word HMMs together.

1.3.2 Pronunciation Dictionary

A pronunciation dictionary defines the phoneme constituents for each word in the vocabulary. Fig. 1.2 gives some entries of a sample dictionary. Here multiple pronunciations will be regarded as having equal a prior probability.

…
ABLY ey b l iy
ABNER ae b n er
ABNORMAL ae b n ow r m ax l
ABNORMALITIES ae b n er m ae l ih t iy z
ABNORMALITIES ae b n ow r m ae l ih t iy z
ABNORMALITY ae b n er m ae l ih t iy
ABNORMALITY ae b n ow r m ae l ih t iy
ABNORMALLY ae b n ow r m ax l iy
ABOARD ax b ow r d
ABODE ax b ow d
ABOLISH ax b aa l ih sh
ABOLISHED ax b aa l ih sh t
…

Fig. 1.2 Part of a sample dictionary

1.3.3 PDTs State Tying

Speech is a very complex signal with many variation factors. Co-articulation is one of the most common variation factors in speech. It means that acoustic realization of a phoneme will be changed with the articulations of neighboring phonemes. To describe the co-articulation phenomena, context-dependent (CD) phonemic HMM is usually used in continuous speech recognition. The most commonly used CD HMM model is triphone, which considers only one left and one right context of a monophone. Different triphones with the same kernel phone are called allophones [3]. For example, *th-ih+nx* (which means the left context is *th* and the right context is *nx*), *l-ih+t*, *uw-ih+nx*, ……, are called the allophones of monophone *ih*. The total number of triphones is a large one considering the commonly used monophone set of English including about 40 phonemes. Training data are always insufficient for reliable estimations of the parameters of HMM models of so many triphones, especially when Gaussian mixtures are used as HMM's output model. Some clustering algorithms such as phonetic decision tree based clustering [10] can be used to reduce the number of physical triphone models to be trained, and a large portion of logical triphone units (such as triphone state) thereafter are shared by or tied to those physical triphone models. Fig. 1.3 is an example of PDT. As illustrated in Fig 1.3, a phonetic decision tree is a binary tree in which a yes/no phonetic question is attached to each node. In phonetic decision tree based state tying, a decision tree is built for each phone state from the context-dependent data of that phone. At the root node of a tree, all allophone states are tied and modeled by one density function. At each node, the state set is tentatively split into two subsets by asking a phonetic context question, the likelihood increment due to the split is measured for each question, and the question that leads to the maximum likelihood gain is chosen and the state split is determined accordingly. This procedure is carried out top-down recursively until one of two termination criteria is met. One is that the data count at one node is less than a predefined threshold, which ensure that all leaf nodes have sufficient data to train a reliable model. Another one is that the

maximum likelihood gain should be larger than another threshold, otherwise the split will stop at this node. Leaf nodes with different parents could be merged if the likelihood loss due to the merging is less than the predefined threshold. Compared with other clustering methods like *k*-means, PDT effectively incorporates acoustic phonetic knowledge of the target language into the clustered models, and it can also model triphone units which do not occur in training data.

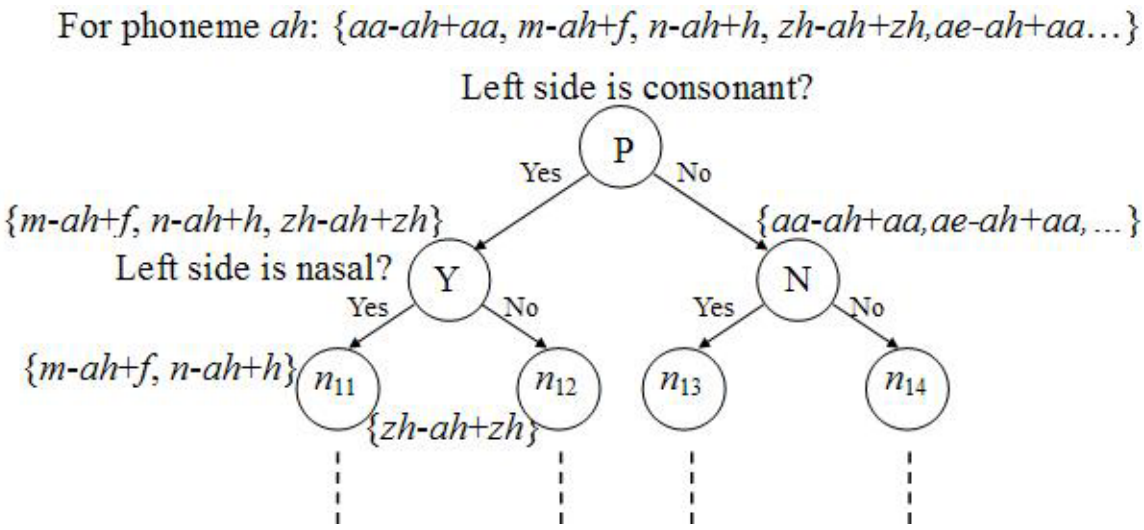


Fig. 1.3 An example of PDT for phoneme *ah*.

1.4 Language Model

Language model (LM) is to deal with this problem: given a sequence of previously spoken words, what is the probability that the word *w* will be spoken next? There are different ways to attack this problem, including Context-Free-Grammar (CFG) [11], Stochastic CFG [12], and *N*-gram model [13]. CFG-based methods try to use a set of knowledge-based rules to define the production of a sentence in words, and *N*-gram LM uses a set of counting-based probabilities to predict the next word. Although the CFG methods seem more reasonable and closer to grammar rules, the *N*-gram method is much more successful in reality since the knowledge-based rules are too complex to be represented by a grammar model such as CFG, whereas *N*-gram LM can be easily

obtained and consistently integrated with acoustic model in a statistical framework based on HMM.

An *N*-gram language model can be simply represented by $p(w_n \mid w_{n-N+1}, \cdots, w_{n-1})$, where $w_{n-N+1}, \cdots, w_{n-1}$ are *N*-1 words that appeared immediately before the current word w_n. By using the fundamental theorem of probability, *N*-gram probability can be easily estimated using a large text corpus. For those words appeared not frequently enough in the corpus, some smoothing techniques are needed to obtain as confident estimations as those common words. Backing-off model [14] is one of the most commonly used smoothing techniques. The idea is to use low-order *N*-gram to approximate the high-order *N*-gram probabilities of those uncommon words. In this model the *N*-gram probability is expressed as follows:

$$p(w_n \mid w_{n-N+1}, \cdots, w_{n-1}) = \begin{cases} \dfrac{count(w_{n-N+1}, \cdots, w_{n-1}, w_n)}{count(w_{n-N+1}, \cdots, w_{n-1})}, & if \quad the \quad string \quad w_{n-N+1}, \cdots, \\ & w_{n-1}, w_n \quad appears \quad frequently \quad enough \\ \alpha(w_{n-N+1}, \cdots, w_{n-1}) p(w_n \mid w_{n-N+2}, \cdots, w_{n-1}), & otherwise \end{cases} \tag{1.7}$$

where $\alpha(w_{n-N+1}, \cdots, w_{n-1})$ is referred to as the back-off coefficient of *N*-1—gram $p(w_n \mid w_{n-N+2}, \cdots, w_{n-1})$, and its main usage is to make the total probability mass of the *N*-gram equals 1 (which would also require discounting the large counts, in general).

The most commonly used *N*-grams are bigram and trigram, where *N* equals 2 and 3 respectively. Higher order *N*-gram such as 4-gram are also used in some research systems.

1.5 Viterbi Time-Synchronous Decoding Engine

As we mentioned before, the purpose of speech recognition is to find the word sequence *W* for a feature sequence *O* that maximizes the posteriori probability *p(W|O)*. The search

component of speech recognition is also called decoding engine, which combines all knowledge sources including acoustic models, language models, pronunciation models, decoding algorithm and determines the overall performances of a speech recognition system.

Viterbi algorithm is one of the most successful decoding algorithms, which is based on the Dynamic Programming (DP) algorithm [2]. Dynamic Programming is a sequential optimization algorithm that decomposes a problem into some sequential, independent sub-problems, and by recursively solving those sub-problems, it will obtain the final solution of the original problem in a bottom-up way. For example, in speech recognition, the optimal word sequence in a sentence from time 1 to time *t* is obtained by recursively getting the optimal word sequence of HMMs from time 1 to *t*-1 until *t* reaches one. In word-conditioned search algorithm, Dynamic Programming is modified to recursively find the best previous word for the current time *t*. The decoding procedure can be decomposed into two steps:

Forward-extension: All possible paths are extended from time 0 to time *T*-1 where *T* is the number of acoustic vectors in a sentence. During the extension, path scores are accumulated by combining the acoustic score and language score for all acoustic vectors up to current frame, and at each time each path will record its best previous word when a new word is created. Different heuristic pruning and lookahead methods may be used at this step to cut off those unpromising search paths so as to speed up the decoding.

Backtrace: After the last frame at *T*-1 has been processed, a best path with the highest score is selected, and then a backtrace is conducted by recursively getting the best previous word of current word recorded in the forward-extension step.

Viterbi algorithm utilizes time-synchronous search, which means that at each time interval, the path scores for different hypotheses are computed based on the same acoustic data, and so heuristic pruning can be easily operated at each time interval, e.g., by comparing those different path scores and keeping only the most promising ones. Beam

pruning is one of the most commonly used heuristic pruning approaches, where a beam value is set as the maximal allowed difference between the highest path score and other competing path scores, so that all paths falling beyond this beam will be pruned out. For LVCSR, experimental results show that only a small percentage of the entire search space (the beam) needs to be kept for each time interval without increasing error rates. According to different level of knowledge sources such as HMM states, HMM phones, and words, it is possible to set different beams for fast decoding. A detailed mathematical formula of Viterbi time-synchronous beam search can be found in [2].

Besides the unique word sequence (best path) output by decoding engine, multiple hypotheses can also be found in decoding search, which are known as *N*-best lists and word lattices.

***N*-best lists:** An N-best list is simply the list of the *N* most likely hypotheses ranked by their scores according to an initial recognition search pass. In general, for each hypothesis there is a score consisting of an acoustic likelihood score and a language model probability resulting from the initial search. In many applications it is desirable to have the *N* most likely hypotheses. For instance, we may want to reprocess the data by using more refined models whose complexity could not allow them to be used directly in the initial recognition search pass.

Word lattices: Word lattices are used by most speech recognizers as a compact intermediate representation of alternative hypotheses and a lattice contains orders of magnitude more hypotheses than an *N*-best list. A lattice is a directed acyclic graph which contains the paths through the search space that are considered probable by the models used in the initial recognition pass. Each link is labeled with a word which is hypothesized between its nodes and the likelihood that the word is uttered in that particular interval. Most lattices have also time information associated with the start and end node of each link. The link likelihoods are obtained as a combination of the acoustic and language model probabilities which are conditioned on the history defined by the

start node of the link. The log-likelihood of a path in the lattice is obtained by adding the log-likelihoods of the links along the path. One method for efficiently constructing word lattices is an extension of the one-pass beam strategy using word dependent copies of the word models or lexical trees [15].

The use of word lattices has become very popular in large vocabulary speech recognition due to the need for incorporating more detailed acoustic models, more complex language models and larger vocabularies, factors that can increase the search space by a few orders of magnitude. The main motivation for generating these word lattices is to provide alternative word hypotheses in different time intervals for speech signals so as to facilitate the integration of more elaborate knowledge sources in this narrowed-down search space.

1.6 Confusion Network and Confidence Annotation

1.6.1 Confusion Network

Confusion network was first proposed in [16], which aligns a word lattice and transforms it into a linear graph in which all paths pass through all nodes. An example of confusion network is shown in Fig. 1.4, where each pair of adjacent nodes defines a position in the CN.

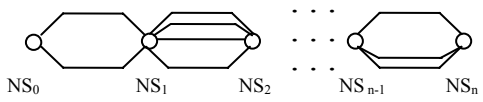


Fig. 1.4 An example of CN

The transformation is performed by a clustering procedure that groups time overlapped links into clusters based on their phonetic similarity and word probabilities while preserving the precedence order of the links encoded in the original lattice. The confusion network was utilized in [16] to generate word hypothesis sequence that minimizes

expected word error rather than sentence error, where the word error criterion has a better match with the Levenshtein-distance based performance metric in speech recognition. Given the alignment and link posterior probabilities of a confusion network, the hypothesis with the lowest expected word error is obtained by picking the word with the highest posterior probability at each position in the alignment.

Besides minimizing expected word error rate and improving recognition performance, CN can also be used in confidence annotation, combining multiple complementary recognition results [17], machine translation, word spotting, handwriting recognition [18], and so on.

1.6.2 Confidence Annotation

With the rising number of different application areas for speech recognition technology, the demand for the ability to spot erroneous words from ASR also increases. In this context confidence annotation can be used to label individual words in the output of the speech recognition system with either correct or incorrect, thus enabling the system and subsequent modules to spot the position of possible errors in the output automatically. In automatic inquiry systems, confidence annotation can be used to avoid unnecessary and very often annoying verification turns if the confidence for the relevant keywords in the speaker utterance is high enough. Confidence annotation can also be applied to unsupervised training and adaptation algorithms. In telemedicine automatic captioning system [19], confidence annotation is used to assist the communication between the doctors and the patients with hearing impairment.

In order to achieve accurate confidence annotation, effective features and classifiers for discrimination between correctly and incorrectly recognized words should be provided. Word posterior probability is one of the important features used in confidence annotation [20][21][22]. Word posterior probabilities obtained in CN [16] are drawing more attentions in recent years, where not only words in the best path but also words in

competing paths are used in computing the probabilities. Besides word posterior probability, many speech decoder-based features have been proposed for confidence annotation, such as acoustic-model score, language-model score, language-model type, local word posterior probability based on state posterior probability [23], number of syllables of word, duration of word and so on. Task-specific features have also been proposed, such as parser-based features in dialog system [24], semantic features in communicator system [25] and so on. Several classification techniques have been proposed for confidence annotation, for example, decision tree and Support Vector Machine (SVM) [22][24].

1.7 Complexity of Automatic Speech Recognition

One feature that is used to characterize the complexity of a speech recognition system is whether the task is to recognize continuous speech or isolated words. In continuous speech it is difficult to determine the boundaries of the words, and there is a larger variability of the acoustic patterns of the words due to the influence of the neighboring word context. Isolated word speech recognition systems do not have this problem since there are distinct silence between neighboring words. Another important feature that affects the complexity of a system is the size of the vocabulary. As the size of the vocabulary increases, there is a greater variability in the acoustic patterns associated with each speech sound unit, and the confusability of the vocabulary words increases. Speaking style is another element of task difficulty. For example, spontaneous, conversational speech is very hard to recognize since it may contain various amounts of filled pauses, repetitions, noises, and even errors. And relatively, speech that is read from texts is easy. Furthermore, exhaustive search is unacceptable in large vocabulary continuous speech recognition, and serious efforts should be made to reduce search space and improve search efficiency so as to produce recognition results in a reasonable amount of time.

The rest of the dissertation is organized as follows. Chapter 2 summarizes the motivations and contributions of this dissertation work. Chapter 3 introduces the fast confusion network algorithm. Random Forests-based word confidence annotation is presented in chapter 4. Improvements in decoding speed and latency are described in chapter 5. Chapter 6 describes the novel lookahead methods for decision tree state tying, and chapter 7 presents Random Forests decision tree state tying. In chapter 8, evaluation experiments and results are described for the techniques proposed. Conclusions are made in chapter 9.

Chapter 2

Motivation and Contribution

My research aims at one of the most challenging tasks of speech recognition, i.e., online large vocabulary conversational speech recognition (LVCSR), with the main goal of developing voice-driven automatic captioning system for Telemedicine, and improving the overall performance of the LVCSR system. In particular, the following scientific and technical problems are addressed:

- Confusion network (CN) can minimize expected word error rate and improve recognition performance, and it can also be used in confidence annotation and many other scenarios. But the algorithm proposed previously [16] for generating a CN is very slow. In this dissertation work we develop a fast CN generation algorithm with linear time complexity O(T), which is capable of transforming a very large word lattice into a confusion network with insignificant time. We further extend the confusion network concept to incorporate the case that a long word is split into short words.
- Confidence annotation plays an important role in automatic speech recognition. Different confidence features and classification techniques have been proposed before. In this dissertation work we develop a confidence annotation method for Telehealth automatic captioning, where we introduce a set of new features from confusion network and statistical significance test, and propose using Random Forests (RFs) as a confidence classifier . The new features are combined with a set of eight confidence features proposed previously, and the Random Forests is compared with Decision Tree and Support Vector Machine.
- Decoding engine is the core of an automatic speech recognition system. A practical LVCSR system needs high accuracy, fast speed, and also acceptable

latency. Improving decoding speed and latency for Telemedicine captioning is investigated in this work. Complementary local word confidence scores are used to prune uncompetitive paths in decoding search. Subspace distribution clustering hidden Markov modeling (SDCHMM) is used for fast generation of acoustic and local confidence scores. We also propose to use pre-backtrace based on detection of prosodic boundaries defined by unfilled pauses, filled pauses, as well as pitch contour to decrease latency in online captioning system.

- Phonetic decision trees (PDTs) play an important role in acoustic modeling. We present two new methods to construct phonetic decision trees (PDTs) for acoustic model state tying, which include constrained lookahead, and stochastic full lookahead methods. Constrained lookahead method finds optimal question among *n* pre-selected questions to decrease effects of outliers, and it also discounts the contributions of likelihood gains by deeper decedents. Stochastic full lookahead method uses sub-tree size instead of likelihood gain as a measure for phonetic question selection, in order to produce small trees with better generalization capability and consistent with training data.
- Most of previous works to improve PDT state tying aim to find one sub-optimal PDT for each phone or phone state. Recently using an ensemble of classifiers as an alternative approach to design a single strong classifier has been studied and advocated [26]. In this work we propose to use Random Forests to train a set of PDTs for each speech unit, and obtain multiple acoustic models accordingly. We investigate several methods of combining the acoustic scores from the multiple models in decoding search. Since computing acoustic scores from the multiple models slow down decoding search, we propose clustering methods to compact the RF generated acoustic models to reduce the time needed for computing acoustic scores.

Chapter 3

Improved Confusion Network Algorithm

CN is becoming a very useful component in speech recognition system. Since its introduction in 1999 [16], CN is attracting more and more attentions. However, the time complexity of the algorithm proposed in [16], referred to as MBS-CN, is high. For a word lattice with T links, the time complexity is $O(T^3)$. Since in an online system like Telemedicine automatic captioning system, the overall processing time, including front-end processing, feature analysis, lattice generation, hypothesis alignment and best hypothesis extraction, needs to be less than 1.0×real time, it is desirable to investigate more efficient methods for confusion network generation.

In this chapter we propose a novel confusion network generation algorithm with linear time complexity $O(T)$. The proposed algorithm is capable of transforming a very large lattice into a confusion network with insignificant time. We further extend the confusion network concept to accommodate the case that a long word gets split into short words.

3.1 Fast Confusion Network Algorithm

The nodes in confusion network, e.g., in Fig. 2.4, are in fact node sets, with each node set including a set of nodes in the original lattice. It is desired to divide the nodes in the lattice into a finite number of sets N_0, N_1, …, N_n, so that the start and end nodes of each link in the original lattice are put into two consecutive node sets. Strictly, not all DAGs can be transformed into confusion network as defined. In order to generate CN with $O(T)$, we resort to the following heuristic approach.

Assumptions:

Let $N=\{n_0, n_1, \ldots\}$ be the set of nodes and $E=\{e_0, e_1, \ldots\}$ be the set of links in the original lattice, where every node $n_i \in N$ has a time mark $t(n_i)$. Let $e_{u->v}$ denote a link in the lattice with the start and end nodes u and v. Let $NS=\{N_0, N_1, \ldots\}$ be the set of node sets in the confusion network and let $E_{N_i->N_j}$ be the set of links with start and end node sets N_i and N_j. The following assumptions are made for the properties of the confusion network:

a. $\forall\, n_i \in N_i$ and $n_j \in N_i$, $t(n_i)<t(n_j) \Rightarrow\ i \leq j$

b. $\forall\, n_i \in N_i$ and $n_j \in N_i$, $t(n_i)=t(n_j) \Rightarrow\ i=j$

c. $\forall\, e_{u->v} \in E$, if for $u \in N_i$ and $v \in N_j$, $e_{u->v}$ corresponds to a link set $E_{N_m->N_n}$, then $i \leq m<n \leq j$ (here $n=m+1$, N_i and N_j are not consecutive, so we should align $e_{u->v}$ to two consecutive node set N_m and N_n.

Algorithm description:

As in the MBS-CN algorithm, the fast CN algorithm requires calculation of the posterior probability for each link in the lattice. The link posterior probability $p(l|O)$ (here O is a sequence of acoustic feature vectors in an utterance as defined before) is defined as the sum of the probabilities of all paths passing through the link l normalized by the sum of probabilities of all paths, which can be computed by the forword-backword algorithm[18]. The fast algorithm consists of the following steps.

Step-1 Compute link posterior probability for each link in the lattice;

Step-2 Sort all nodes in N in order of increasing $t(n_i)$;

Step-3 Assign n_0 to N_0;

Step-4 For each node n_i, in the order of $i=1, 2, \ldots,$

i) Suppose n_{i-1} belongs to N_j. If there is no link between any node in N_j and n_i, assign n_i to N_j, otherwise assign n_i to N_{j+1}.

ii) For every link $e_{u->n_i} \in E$, suppose u belongs to N_s and n_i belongs to N_t. If $t = s+1$, then the link is directly assigned to $E_{N_s->N_t}$. Otherwise, the link is assigned to

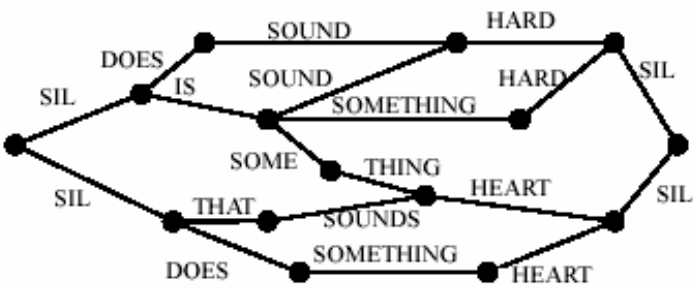
$E_{N_{n-1}->N_n}$ with $s+1 \le n \le t$, where the link word probability and degree of time overlap are considered in the link placement, i.e., $n = \arg\max_{s+1 \le k \le t} \{sim(E_{N_{k-1}->N_k}, e)\}$

with $\text{sim}(E_{N_{k-1}->N_k}, \text{e}) = \frac{1}{|E_{N_{k-1}->N_k}|} \times \sum_{l \in E_{N_{k-1}->N_k}} sim(w(l), w(e)) overlap(E_{N_{k-1}->N_k}, e)$

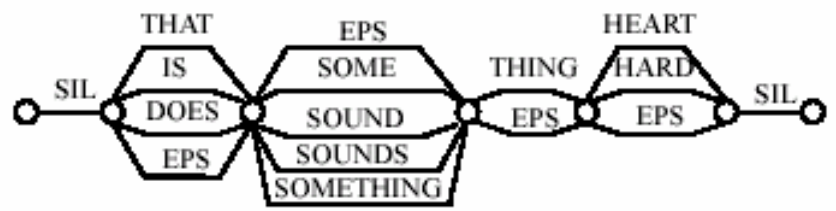
where $w(l)$ and $w(\text{e})$ are words corresponding to l and e, $sim(.,.)$ is the phonetic similarity between two words, computed from the most likely phonetic base forms, $Overlap(E_{N_{k-1}->N_k}, \text{e})$ is defined as the time overlap between $E_{N_{k-1}->N_k}$ and e normalized by the sum of their lengths. The length of $E_{N_{k-1}->N_k}$ is $T_{max}(N_{k-1}) - T_{min}(N_k)$, where $T_{max}(N_i) = max\{t(n_j): n_j \in N_i\}$ and $T_{min}(N_i) = min\{t(n_j): n_j \in N_i\}$

The time complexity of step 4 is based on the number of links in the set $E_{N_s->N_t}$. In general, even for a very large lattice, this number is less than 100. So the time complexity of step 4 is $O(T)$. Furthermore, word lattices as generated by various speech decoder engines, including HTK, have the nodes naturally sorted in order of increasing $t(n_i)$. Therefore, the time complexity of the proposed confusion network generation algorithm is $O(T)$.

An example of transforming a word lattice into a confusion network by the proposed algorithm is shown in Fig. 3.1. In Fig. 3.1, *EPS* represents a *NULL* link, and its posterior probability equals 1 minus the sum of the posterior probabilities of the rest links at this position.



(a) Initial word lattice



(b) Confusion network

Fig. 3.1 Transform a word lattice into a confusion network

3.2 Extended Confusion Network Algorithm

In a confusion network, each link sits between two consecutive node sets. If a link is very long, however, aligning the link to two consecutive node sets may not be reasonable. Fig. 3.2 illustrates such a case, where *JULY* is a much longer word than either the word *DO* or *I*.

Fig. 3.2 An example of a long word broken into two short words

After forced alignment, one of the following two results will occur in the confusion network:

Fig. 3.3 Two possible alignments in confusion network for the case of Fig. 3.2

Based on the pronunciation and time information of these three words, we know that *JULY* should correspond to a concatenation of *DO* and *I*, that is, on one path, the word output is *JULY*, and on another path the word output should be *DO* followed by *I*, which is a typical case of splitting a long word into two short words. To accomodate such a scenario and overcome the limitation in confusion network, we modify the algorithm

such that when links are aligned, one link can end at two nonconsecutive node sets N_i and N_{i+2}. The modified confusion network is illustrated in Fig.3.4.

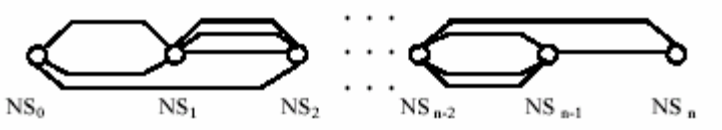


Fig. 3.4 An example of extended confusion network

If the case of Fig. 3.5 occurs, we hold the link with higher posterior probability and align the other one between two consecutive node sets.

Fig. 3.5 The case prohibited in extended confusion network

In some cases, a long word may be split into three or more short words. For simplicity, our algorithm only allows the case of one long word split to two short words.

The modification of the algorithm can be made in sub-step ii) of step 4. If $t>s+1$, the link e should be put in either $E_{N_{n-1}->N_n}$ or in $E_{N_{n-2}->N_n}$, based on link word probability and degree of time overlap. Computation of *sim*(., .) and *overlap*(., .) remain the same as before.

By applying the improved alignment algorithm, the lattice in Fig. 3.1 (a) is transformed into the modified confusion network in Fig. 3.6.

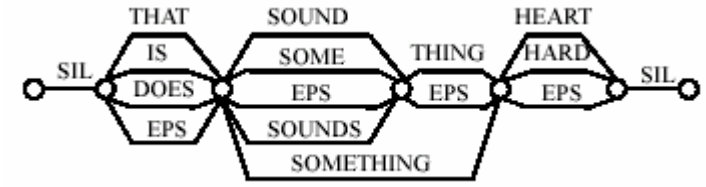


Fig. 3.6 Extended confusion network for the case of Fig. 3.1 (a)

Chapter 4

Random Forests-based Confidence Annotation

As we mentioned before, confidence annotation plays an important role in automatic speech recognition. In order to achieve accurate confidence annotation, effective features and classifiers for discrimination between correctly and incorrectly recognized words should be provided. We propose a novel method for confidence annotation in Telemedicine automatic captioning. We present two types of features: decoder-based features and confusion network (CN) based features. We also propose to use Random Forests to classify the recognized words as being correct or wrong.

4.1 NOVEL FEATURES

In this section we introduce four novel features for confidence annotation, where three are based on CN and one is from acoustic model [17].

4.1.1 Entropy for CN

Confusion network is used to generate word sequence hypothesis that minimizes expected word error rate. Given the alignment and the link posterior probabilities of a confusion network, the word sequence hypothesis with the lowest expected word error is obtained by picking the word with the highest posterior probability at each position in the alignment. The word posterior probability included in the confusion network is a good confidence feature. Besides word posterior probabilities, here we consider the entropy of words for each position of CN based on the word posterior probabilities derived for CN. Entropy measures the difference of word posterior probabilities among words in the same position in the CN, and ambiguity of word identity can be better captured by entropy than the word posterior probability alone. Entropy for CN is defined as:

$$En(w \mid O) = -\sum_{i=1\ldots m} p(w_i \mid O) \log p(w_i \mid O) \tag{4.1}$$

where w is the word in the recognition output, w_i, $i = 1 \ldots m$ are the words that are in the same position with w in CN, and w is one of w_i's.

4.1.2 Bigram, Trigram Posterior Probabilities in CN

Similar to word posterior probability, bigram and trigram posterior probabilities are also word-level posterior probabilities in CN, but these two are conditional probabilities given the context in the recognition output, which are respectively defined as following.

$$p(w_i \mid w_{i-1}, O) = \frac{p(w_i, w_{i-1} \mid O)}{p(w_{i-1} \mid O)} \tag{4.2}$$

$$p(w_i \mid w_{i-1}, w_{i-2}, O) = \frac{p(w_i, w_{i-1}, w_{i-2} \mid O)}{p(w_{i-1}, w_{i-2} \mid O)} \tag{4.3}$$

where w_{i-2}, w_{i-1}, w_i are consecutive words in the recognition output w_1, … , w_{i-2}, w_{i-1}, w_i, …, w_n. The joint posterior probability $p(w_i, w_{i-1} \mid O)$ and $p(w_i, w_{i-1}, w_{i-2} \mid O)$ can be computed using forward-backward algorithm in word lattice in a similar way as the single word posterior probability $p(w_i \mid O)$.

4.1.3 Accumulated probability

For a class with one-dimension Gaussian distribution $N(\mu_0, \sigma^2)$, given an observation x, the accumulated probability (AP) of x in this class is defined as $AP(x) = \Pr ob(|X - \mu| > |x - \mu_0|)$, which is the shaded area in Fig. 4.1.

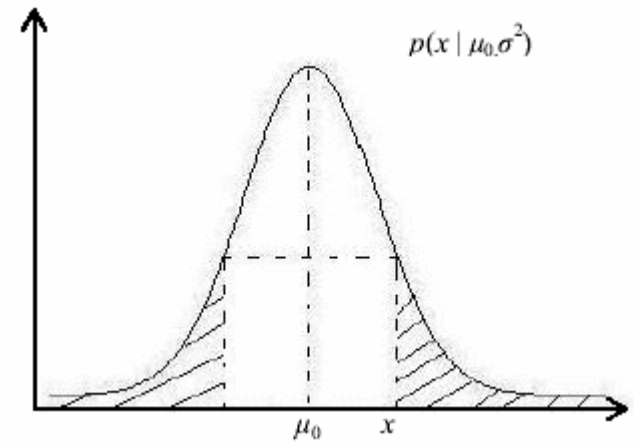


Fig. 4.1 Accumulated probability for a Gaussian distribution

The larger $AP(x)$ is, the higher confidence we will have that x belongs to this class

For classification, we need to test if an observation x belongs to some class C. If the class-conditional distribution is a Gaussian distribution $N(\mu_0, \sigma^2)$, we can use $AP(x)$ as a confidence feature. Compared with acoustic likelihood score, accumulated probability captures information of distribution spreadness more effectively.

In speech recognition, tied state is modeled by a multivariate Gaussian mixture distribution and diagonal co-variance matrix is often used. We therefore define the accumulated probability $AP_i(\underline{x})$ for each n-dimensional Gaussian distribution as the product of the accumulated probabilities in individual dimensions. The accumulated probability of an n-dimension Gaussian mixture distribution is then defined as

$$AP(\underline{x}) = \max_i (w_i AP_i(\underline{x})) \tag{4.4}$$

or

$$AP(\underline{x}) = \sum_{i=1...M} w_i AP_i(\underline{x}) \tag{4.5}$$

where w_i is the ith mixture weight. Our empirical evaluation showed that there is no significant difference between these two definitions.

For a word w in the recognition results, denote x_t, $t = 1,2,\ldots,T$ as the corresponding acoustic observation sequence and M_t, $t = 1,2,\ldots,T$ as the corresponding model sequence. We then define the log AP for word w as

$$\log AP(w) = \sum_{i=1}^{T} \log(AP(x_t \mid M_t)) \tag{4.6}$$

4.2 Random Forests

Random Forests is a classifier based on the algorithm developed by Leo Breiman and Adele Cutler [27] [28]. The classifier uses a large number of decision trees. To classify a new object, the object is sent to each tree in the forest. Each tree gives a classification for the object and the forest chooses the classification having the most votes.

Each tree in the forest is grown as follows. First, choose *N* samples randomly with replacement from the original training dataset for growing the tree. Second, select *m* variables randomly out of a total of *M* variables and use the best split determined by these m variable to split the node. The value of m is held constant during the forest growing. Third, each tree is grown to the largest extent possible, without pruning.

The Random Forests error rate depends on the following two factors. The first is the correlation between any two trees in the forest, where increasing the correlation increase the error rate. The second is the strength of each individual tree in the forest. A tree with a low error rate is a strong classifier, and increasing the strength of the individual trees decreases the forest error rate.

Reducing *m* reduces both correlation and the strength, while increasing it increases both. Therefore a proper value for m is needed. Empirically, *m* is set to be approximately the square root of *M* in the software R, and $\log_2 M$ in [28].

Random Forests is considered unexcelled in accuracy among current classification techniques, and can handle thousands of input variables without variable deletion [27]. Random forests do not over-fit as more trees are added.

Random Forests can provide estimates on which variables being important in classification. The value of importance is defined as the total decrease in node impurities from splitting on the variable, averaged over all trees. The node impurity is measured by

the Gini index. For further details of Random Forests, please see [27] [28].

4.3 Confidence Annotation in Telemedicine

In this section, we describe the confidence annotation method for Telemedicine automatic captioning, including features and classification techniques.

4.3.1 Features

The complete set of features include the novel features we introduced in Section 4.1 and eight other features that were previously proposed in the literature. The features are categorized as decoding engine-based features and CN-based.

Decoding engine-based features

We use eight decoder-based features.

- Acoustic-Score (AS): Acoustic model score of a word.
- Language-Score (LS): Language model score of a word.
- Language-Type: Language model type of a word, backoff or not.
- Total-Score: AS + lmscale*LS, where Lmscale is the weight of language model score used in decoding.
- *AP*: Accumulated probability of a word, introduced in 4.1.3.
- Ave-*AP*: Average *AP*, defined as *P*-value divided by the duration of the word.
- LWPP: Local word posterior score.
- Ave-LWPP: Average LWPP, defined as LWPP divided by the duration of the word.

The feature LWPP was proposed in [23]. To define LWPP, the posterior probability of the state s_i conditioned on the observation x is defined as:

$$p(s_i \mid x) = \frac{p(x \mid s_i)p(s_i)}{p(x)} = \frac{p(x \mid s_i)p(s_i)}{\sum_{s_j \in D} p(x \mid s_j)p(s_j)} \quad (4.7)$$

where D is the set of all states survived after pruning. Assuming equal prior probabilities for all states, the formula is simplified as:

$$p(s_i \mid x) = \frac{p(x \mid s_i)}{\sum_{s_j \in D} p(x \mid s_j)} \quad (4.8)$$

Further assuming for the word w the state sequence and the observation sequence to be $s_m, \ldots s_n$ and $x_m, \ldots x_n$, the LWPP of w is defined as:

$$LWPP(w) = \log[\prod_{i=m\ldots n} p(s_i \mid x_i)] \quad (4.9)$$

CN-based features

- Entropy: Entropy for CN, introduced in 4.1.1.
- Pos-Score: Word posterior score in CN.
- Bi-Pos: Bigram word posterior score from CN, introduced in 4.1.2.
- Tri-Pos: Trigram word posterior score from CN, introduced in 4.1.2.

4.3.2 Classification Techniques

Three classification techniques were investigated: Random Forests, Decision Tree and SVM. The statistical software R [29] was used in constructing the three classifiers. For Random Forests, we also examined the ranks of the features based on their importance in the confidence annotation.

Chapter 5

New Improvements in Decoding Speed and Latency in Online Captioning System

5.1 Introduction

Decoding time efficiency has been an important issue in ASR systems. Usually, the time efficiency of a decoding engine is measured by the criterion of real-time factor. However, for interactive online conversational systems, such as automatic captioning system in Telemedicine [19], a system needs to provide other functions in addition to decoding, like speech stream separation, speaking rate estimation, confidence annotation, and so on. Since the overall processing time should be less than 1.0×real time, in such a system, a real-time decoding engine is not fast enough.

Decoding can be viewed as a search of an optimal path in a huge search network. An exhaustive search through the network is intractable in LVCSR [30]. To make search feasible, the search space needs to be reduced by pruning. Beam pruning is probably the most important pruning criteria used in LVCSR decoders, which retains only paths with likelihood scores deviate from the score of the best partial path hypothesis within a beam width [31]. Another common pruning criterion is histogram pruning, which limits the number of active paths by retaining only a predefined number of best paths. Besides of beam and histogram prunings, quickly computing acoustic score and accessing language score are also important ways to speed up decoding.

Dynamic programming based decoding algorithms utilize backtracking and determine the best path, i.e., the recognition result, until reaching the end of the input. So the latency, i.e., the time lapse from a speaker starting to talk till recognition result being outputted, is at least the duration of the utterance. In Telemedicine, doctors can talk

without a noticeable break for tens of seconds. Such a long utterance contains quite a few sentences and sometimes with many filled pauses. Our speech stream separation module [19] can detect low-energy un-filled pauses to break speech inputs into utterances, however, excessively long utterances remain in the separation output. Such long inputs to a decoder reduce decoding speed and increase latency, since as the inputs get longer, more search paths will be expanded and more memory will be consumed, and the time spent in waiting for recognition outputs will be increased. As the consequence, conversations between doctor and patient become sluggish.

Our decoding engine performs large vocabulary continuous speech recognition via one-pass time-synchronous Viterbi beam search with a tree-copy based search organization [32]. In this chapter we propose three methods to speed up the decoding engine and decrease its latency. First, we propose to use complementary local word confidence scores to prune unpromising candidate paths during search, based on local word posterior probability score and word *P*-value. Second, subspace distribution clustering hidden Markov modeling (SDCHMM) [33] is used to speed up the generation of acoustic scores and local word confidence scores. In Gaussian density clustering for SDCHMM, we propose to use overlap accumulative probability (OAP) to measure the similarity of Gaussian *pdf*'s. Third, in order to decrease latency, we propose to use pre-backtrace to output recognition results incrementally prior to reaching the end of input by using both energy and pitch as prosodic cues for pre-backtrace. It is worth noting that unlike the read speech problem dealt with in [34], where low energy pauses were used as cues for pre-backtrace, in conversational speech, energy cue alone is insufficient and therefore both energy and pitch are used.

5.2 Confidence-based Pruning

Confidence-based pruning (CBP) was proposed in [35] to guide the search for most promising paths in addition to beam and histogram prunings. In this technique,

confidence scores play the role of an online filter that is applied to the word level partial hypotheses to aid the decision of whether to consider them for further path expansions or to discard them from the search space. In [35], only the feature of local word posterior probability (LWPP) was used. Here we use two complementary confidence scores. One is word level accumulated probability *AP* (*w*), another one is *LWPP*(*w*) . Although *AP(w)* and *LWPP(w)* both provide confidence information for the word *w*, they are different. Accumulated probability captures information of distribution spreadness more effectively, while *LWPP(w)* catches the difference of acoustic scores among the path candidates. A competitive path should have both large *AP* (*w*) and *LWPP(w)*.

In the search process, at every time frame *t*, each word *w* that reaches its final state will be evaluated by confidence scores of *AP*(*w*) and *LWPP(w)*. These two values reflect how well the hypothesized word can account for the acoustic evidence. The better the acoustic evidence is supported by a hypothesized word, the higher the confidence that the word is in fact correct. If *AP*(*w*) or *LWPP(w)* is smaller than their predefined thresholds, the corresponding path will be pruned.

5.3 SDCHMM

5.3.1 Theory of SDCHMM

SDCHMM was first introduced by Bocchieri and Mak [33], with the aim of building compact acoustic models. The theory of SDCHMM is derived from that of CDHMM in which state-observation distribution are estimated as mixture Gaussian densities with *M* components and diagonal covariances. From section 2.3.1 we know that the observation probability of the *j*th state of a CDHMM is given by $b_j(o) = \sum_{m=1}^{M} c_{j,m} N(o; \mu_{jm}, \Sigma_{jm})$ with $\sum_{m=1}^{M} c_{j,m} = 1$.

The key observation is that a Gaussian with diagonal covariance can be expressed

as a product of subspace Gaussians where the subspace (or streams) are orthogonal and together span the original full feature vector space. To derive *K*-stream SDCHMMs from a set of CDHMMs, we first partition the feature set with *D* features into *K* disjoint feature subsets with d_k features with $\sum_{k=1}^{K} d_k = D$. Each of the original full-space Gaussians is projected onto each feature subspace to obtain *K* subspace Gaussians of dimension d_k, $1 \le k \le K$, with diagonal covariances. Thus formula (2.3) can be rewritten as

$$b_j(o) = \sum_{m=1}^{M} c_{j,m} \left(\prod_{k=1}^{K} N(o_k; \mu_{jmk}, \Sigma_{jmk}) \right) \tag{5.1}$$

where o_k, μ_{jmk}, and Σ_{jmk} are the projection of the observation *o*, and mean and variance vectors of the *m*th mixture component of the *j*th state onto the *k*th subspace, respectively.

For each stream, we tie the subspace Gaussians across all states of all CDHMM acoustic models. Hence the state observation probability in (5.1) is modified as

$$b_j^{SDCHMM}(o) = \sum_{m=1}^{M} c_{j,m} \left(\prod_{k=1}^{K} N^{tied}(o_k; \mu_{jmk}, \Sigma_{jmk}) \right) \tag{5.2}$$

In order to get SDCHMMs, we first train CDHMMs for all the phonetic units, where state observation distributiona are mixture Gaussian densities with diagonal covariances, and then convert the CDHMMs to SDCHMMs by tying the subspace Gaussians in each stream. Details of the stream definitions and clustering algorithms can be found in [33].

5.3.2 SDCHMM in Telemedicine

Since in LVCSR, as used in Telemedicine automatic captioning system, the number of acoustic models is very large, computing acoustic scores and *AP* consume a lot of time, especially *AP*, where the time needed to compute a *AP* is several times more than that for computing an acoustic score. Although we use table-lookup to speed up the computation of *AP*, prior to lookup we need to normalize the observation x_t for each model to be zero

mean and unit standard deviation, which is a heavy computation load due to the large number of acoustic models. If SDCHMM is used, then the Gaussian density scores of all continuous density hidden Markov models (CDHMMs) can be approximated by certain combinations of a small number of subspace distribution prototypes. In the decoding process, all these subspace Gaussian log likelihoods and log *APs* can be pre-computed once for each dimension at the beginning of each frame, and their values are stored in lookup tables. Then for each Multivariate Gaussian in a physical model, the log likelihoods and log *AP*s can be computed as the summation of pre-computed values in different dimensions together with the log mixture weight. So SDCHMM is computationally efficient.

Here we define each feature dimension as a stream in SDCHMM, and to convert CDHMM to SDCHMM, a clustering of Gaussian densities in each feature dimension is made and therefore a proper measure of similarities of the *pdf*'s is needed. The classification-based Bhattacharyya distance (CBB) was used in [33]. Li et al proposed Kullback-Leibler divergence (KLD) for SDCHMM [36]. For Gaussian distributions, it was found that KLD overstates the difference between Gaussian densities when they have very different variances [37].

OAP was first proposed in [37], where it was used for Frame Discrimination (FD) training. Here we use OAP as the similarity measure for clustering a Gaussian *pdf f* to a Gaussian prototype *g*, which is defined as:

$$S(f,g)=\int_{-\infty}^{+\infty}\min(f,g)dx \tag{5.3}$$

The value *S(f, g)* is between 0 and 1. If *f* and *g* are the same, then *S(f, g)* equals 1. For two Gaussian *pdf*s, S(·,·) equals one of the shaded areas in Fig. 5.1:

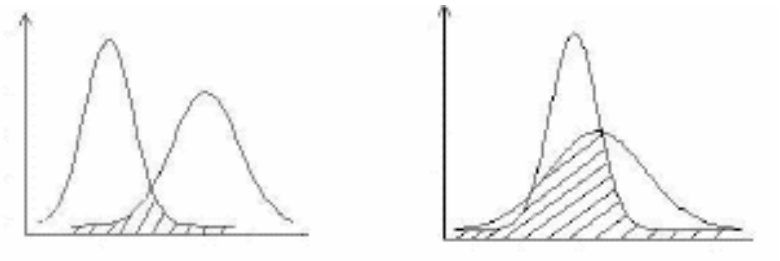


Fig. 5.1 Definition of OAP between two Gaussian *pdf*s

The optimal prototype, $g(\mu_p, \sigma_p^2)$, of a cluster of N Gaussian distributions, $f(\mu_i, \sigma_i^2)$, $i = 1 \ldots N$, is obtained by the following formulas:

$$\mu_p = \frac{1}{N}\sum_{i=1}^{N}\mu_i \tag{5.4}$$

$$\sigma_p^2 = \frac{1}{N}\sum_{i=1}^{N}\{\sigma_i^2 + (\mu_i - \mu_p)^2\} \tag{5.5}$$

To initialize the iterative *k*-means clustering procedure for converting CDHMMs to SDCHMMs with K subspace Gaussian prototypes per dimension, we first split all the Gaussians into K subspaces, based on the mean values of the Gaussians. In each dimension, we order all the M Gaussians to be clustered with the increasing order of $\mu_i's$. We assign the first M/K Gaussians to cluster 1, the second M/K Gaussians to cluster 2, and so on, and compute the Gaussian prototype of each cluster with formulas (5.4) and (5.5). Subsequently, the algorithm iterates between cluster assignment and prototype update. In cluster assignment, each Gaussian is assigned to the cluster producing maximal S(·,·). In prototype update, the formulas (5.4) and (5.5) are used. Usually after a few iterations the Gaussian prototypes will converge.

5.4 Pre-backtrace Decoding

It is widely acknowledged that prosodic features play an important role in the definition and hence the detection of syntactic boundaries. Since most phonological rules are constrained to operate within a phrase, it is reasonable to consider pre-backtrace at

syntactic boundaries which are often associated with prosodic boundaries.

Unfilled pause (or pause) is a well recognized feature of prosodic boundary. In spontaneous speech, on the other hand, filled pause, such as AH, UM, is also a promising feature. Many filled pauses last for sufficient durations and therefore they serve as potentially good places for pre-backtrace. The F_0 contour is a representation of intonation. Dips in F_0 contours often coincide with prosodic boundaries and therefore they can be used in addition to energy for detection of syntactic boundaries. In [38], a rule-based analysis of macro- and microscopic features of F_0 is conducted to find prosodic-syntactic boundaries. The F_0 contour is segmented at dips (minima) in the energy contour of the same speech utterance, and rules are applied to generate candidate syntactic boundaries at dips in the F_0 contour. We detect dips in F_0 contour by following the rules of [38]. We use autocorrelation of residues of linear prediction coding (order 16) to produce F_0. For each speech segment marked by a pair of dips in the energy contour, we fit the pitch contour by a recursive line with the method of least square error. If the error averaged over the segment exceeds a threshold, the segment is divided into two parts at the point where the maximum error occurs and a recursive line is fitted again for each part. The process iterates until we find a recursive line for each segment. All dips in F_0 contour obtained by this procedure are treated as prosodic boundaries.

With the aim of reducing latency in our decoding engine, we detect syntactic boundaries based on detections of short unfilled pauses, filled pauses, and dips in F_0 contour during one-pass speech decoding and use the cues to perform pre-backtrace. In acoustic modeling, one HMM model was trained for short pauses, and several HMM models were trained for different patterns of filled pauses. In decoding, if short pause state rank as the best hypothesis consecutively with a duration exceeding a threshold, then the boundary is accepted; similarly, if the last state of a filled pause HMM ranks as the best hypothesis consecutively for several frames, then the boundary is accepted; finally, if a prosodic boundary is detected at some point due to a dip in F_0 contour, and

the last state of some word ranks as the best hypothesis for several consecutive frames, then the boundary is also accepted. Once a prosodic boundary is accepted, the decoder starts backtracing to find the best partial path and outputs it. Decoding then continues by using the fixed word history that includes the last several words on the best partial path just produced.

Chapter 6

Novel Lookahead and Random Forests Decision Tree State Tying

6.1 Introduction

Phonetic decision tree (PDT) state tying is commonly used in acoustic modeling for large vocabulary continuous speech recognition since it can model triphone units or context which do not occur in training data. In the traditional PDT algorithm [10], each node split is based on local likelihood gain. In this way, the constructed tree is only locally optimal in general. *K*-step lookahead search is a technique for overcoming the limitation of greedy search. When applied to decision tree induction, lookahead method attempts to predict the profitability of a split at a node by estimating its effect on deeper decedents of the node [39]. Since *K*-step lookahead algorithm has an exponential complexity, usually only one- or two-step lookahead is used. Another questionable issue about the traditional PDT algorithm is the use of likelihood gain as the criterion to split a node.

We present two novel lookahead methods of constructing PDTs for acoustic model state tying. The first one is called constrained lookahead. In this method, instead of searching for an optimal question within the whole question set, we first find *n* questions which give the *n*-best local increases in likelihood, and then we constrain the lookahead search for the optimal question to be among the *n* questions to split the node, where the contribution of the deeper decedents is reduced as a function of their levels in the tree [39]. The second method is called stochastic full lookahead, originally proposed in [40] for small tasks in machine learning. In this method, instead of using likelihood gain as a judgment to select a phonetic question for node split, subtree sizes are used to find a compact tree that is consistent with the training data set. Since finding an optimal subtree at each node is a NP-Complete problem, a stochastic method is used to generate an

ensemble of subtrees for each current node split, and the question that minimizes the subtree size is preferred [40]. In both methods we propose to use phone-state dependent threshold of likelihood gain to decide if a node split should go further or not. We also propose to use a fast confusion network (CN) algorithm to combine recognition hypotheses produced by acoustic models resulting from different PDT training methods.

6.2 Phone-state Dependent Threshold

In commonly used method of constructing PDTs, for example HTK [41], a node is split if the largest increase of likelihood score due to the split exceeds a predefined threshold, where the threshold is kept identical for all phone states. Here we propose to use a threshold for likelihood gain that is proportional to the occurrence count of data in each phone state. For a PDT of phone *i* and state *j*, let L_p be the log likelihood score of a parent node to be split, L_y and L_n be the log likelihood scores of the two children nodes, and N_{ij} be the total occurrence count under the root node. We select the question that maximizes the likelihood gain scaled by N_{ij} as

$$\Delta L = \frac{L_y + L_n - L_p}{N_{ij}} \tag{6.1}$$

If the largest ΔL is small than a predefined threshold *C*, then the split will stop, that is

$$\max(\frac{L_y + L_n - L_p}{N_{ij}}) < C \Rightarrow \max(L_y + L_n - L_p) < N_{ij}C = \lambda_{ij} \tag{6.2}$$

We call λ_{ij} the phone-state dependent threshold. Compared to the fixed threshold in commonly used method, the threshold is proportional to the total occurrence count of each phone state, which helps prevent phone states with large amounts of training data to grow overly large trees. The minimum occurrence count threshold of leaf nodes is still kept as a constant to ensure sufficient training data for estimation of observation probability distributions.

6.3 Constrained Lookahead

It is well known that the greedy search method of choosing a phonetic question based on largest likelihood gain in the current node split only leads to a local optimization of the decision tree. It seems plausible that a global optimization on decision tree training would improve acoustic models. Since the computational cost of global optimization is too high, lookahead methods can be used to select the phonetic question that produces largest likelihood gain on its deeper decedents. However, previously proposed lookahead methods did not yield improved performance in speech recognition [39], [42]. Instead it gave worse results than conventional PDT training, mainly due to overtraining.

Here we present a constrained lookahead method to optimize the PDTs. At each node, instead of searching for the optimal question within the whole question set, we first find *n* questions which give the *n*-best local increases in likelihood, and then we find the optimal question among the *n* questions to split the node through a *K*-step lookahead. The n-best constraint is applied to node splits in each lookahead level as well. The rational of the *n*-best approach is that the optimal question is likely one of the *n*-best questions at the current node, since these questions in general have larger impacts on data partition into leaf nodes than questions at decedent nodes. Narrowing the search space may also decrease the effect of outliers. Fig. 6.1 illustrates the construction of a PDT that employs *K*-step lookahead.

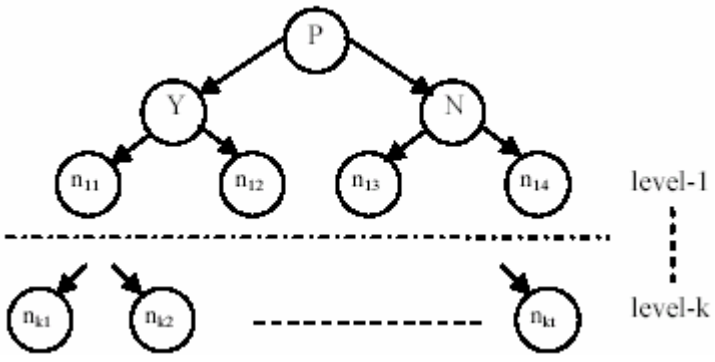


Fig. 6.1 PDT construction using K-step lookahead

It was previously reported that the contribution of likelihood gains by the deeper decedents should be reduced with the decedent level [39]. Here we adopt the idea and define the likelihood gain as a weighted average of likelihood gains at successive levels in the k-step lookahead window. Let ΔL_i be the likelihood increase at the *i*th level, and ΔL_p equals $L_y + L_n - L_p$. At each node, the likelihood gain is defined as

$$\Delta L = \Delta L_p + \sum_{i=1}^{k} \frac{\Delta L_i}{i + tree\_level(P)} \tag{6.3}$$

where *tree_level*(root)=0, and we select the question that produces the largest ΔL.

6.4 Stochastic Full Lookahead

Tree size has been an important issue in constructing PDTs. One obvious concern is that speech decoding speed is dependent on the number of tied states in acoustic models, since as the number of tied states increases, it takes more time for a decoding engine to compute the likelihood scores. The number of tied states is directly decided by the sizes of the PDTs. Another important concern is that a good model should represent the salient clustering structure of data, instead of over fitting the data, and as such compact model is also preferred.

The goal of the stochastic full lookahead method is to find small trees consistent with the training data. In contrast, traditional tree-pruning based methods first grow a large tree by greedy search and then prune off noncontributing leaves or subtrees to reduce the size of the tree, which in general yields small trees that are not consistent with data.

Exhaustive lookahead at each node will definitely lead to the smallest tree, but the computational cost is prohibitively high, and therefore a stochastic approach is used. For each question under consideration at the current node, the stochastic method generates two ensemble of subtrees, one for its left child node and one for its right child node,

respectively. Each ensemble of subtrees provides an estimate of the subtree size. In growing a random subtree, when a phonetic question is used to split a node, the split question is drawn randomly with a probability proportional to the likelihood gain due to splitting the node by this question. Table 6.1 describes the question selection procedure used in the stochastic lookahead method, and Table 6.2 describes the question selection procedure in growing the random subtrees at each node. We modify the algorithm in [40] such that at each node we limit the search space to that defined by the *n*-best questions instead of the whole question set, which decreases the training time greatly. In addition, the n-best approach may yield a good balance between the aspect of model-data fit and that of model compactness.

In determining subtree size, the smallest size of random subtrees in an ensemble is taken, since a random tree by its nonexhaustive search nature can only overestimate the minimum size of subtree [40].

Table 6.1 Question selection procedure in stochastic full lookahead

Get a question set QS including *n* questions which give the *n*-best local increase in likelihood
For each question $q_i \in QS$
Split the node into yes-children and no-children
$\min_{yes} \leftarrow \min_{no} \leftarrow \infty$
Repeat *r* times
Grow a random sub-tree, sub_{yes} of yes-children
$\min_{yes} \leftarrow \min(\min_{yes}, sizeof(sub_{yes}))$
Grow a random sub-tree, sub_{no} of no-children
$\min_{no} \leftarrow \min(\min_{no}, sizeof(sub_{no}))$
$size_i \leftarrow \min_{yes} + \min_{no}$
Return q_i whose $size_i$ is minimal

Table 6.2 Question selection procedure in growing random subtrees

Get a question set QS including n questions which give the n-best local increase in likelihood
$q^* \leftarrow$ Choose question at random from QS; for each question q, the probability of selecting it is proportional to the local likelihood increase for the training data set
Return q^*

6.5 Integration of Recognition Results

Upon obtaining recognition hypotheses by using different acoustic models from different PDT training methods, we put the results together into a simple lattice and then align the lattice into a CN by using the fast CN algorithm. The final hypothesis is obtained by picking words with the highest posterior probabilities at each position in the CN. Fig. 6.2 is a simple lattice constructed from five recognition hypotheses, and Fig. 6.3 is the corresponding CN.

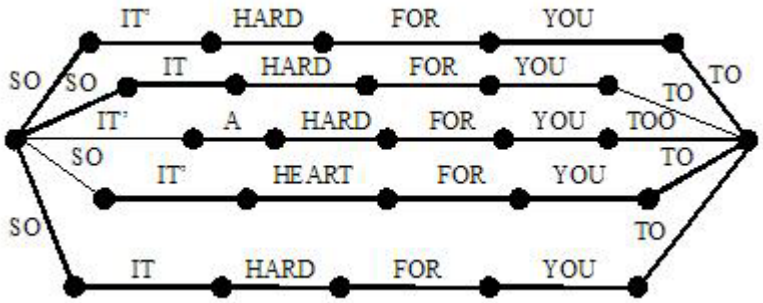


Fig. 6.2 A simple lattice of recognition results.

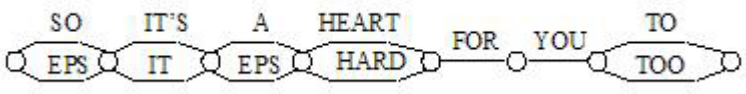


Fig. 6.3 Confusion network for the lattice in Fig. 6.2.

Chapter 7

Random Forests Phonetic Decision Tree state Tying

Many efforts have been made to improve PDT state tying based approach in acoustic modeling [43]-[50]. Most of the previous works, including the ones we introduced in chapter 6, aim to find one sub-optimal PDT for each phone or phone state, since in general, construction of a globally optimal decision tree is an intractable problem. In recent years, using an ensemble of classifiers as an alternative approach to design a single strong classifier has been studied and advocated [26]. The original ensemble method is Bayesian averaging [51], which weights the classifiers by their posterior probabilities. More recent algorithms include Error Correcting Output Coding (ECOC) [52], Bagging [53], and boosting [54]. ECOC prepares redundant discriminant functions and then construct a classifier by combining these discriminant functions. Bagging is a voting method whereby base-classifiers are made different by training them over slightly different training data sets. In boosting, a sequence of complementary base-learners is generated by training the next learner on the mistakes of the previous learners. There are several boosting algorithms, depending on their mathematical forms of the strength and weight applied to data samples. AdaBoost [55] is a popular and historically most significant boosting algorithm. Recently more algorithms such as gradient boosting [55] and LPBoost [56] have also been proposed. Random Forests (RF) generates an ensemble of decision trees by stochastically sampling training data and splitting questions. RF is considered unexcelled in accuracy among current classification techniques [27] [28]. RF was first used for constructing PDTs in [57], where multiple PDTs were grown for each speech unit by randomly selecting a split from the top-K best splits and each PDT generates an acoustic model for the speech unit. Multiple recognition outputs resulting from the multiple acoustic models were combined at the word level.

In this chapter, we present a Random Forests based method to train multiple PDTs in acoustic modeling inspired by the RF algorithm of Breiman and Cutler [27] [28]. Instead of combining recognition results at the word level after decoding search [57], we combine acoustic scores at the model level in decoding procedure to achieve better accuracy performance and faster speed. We generate RF tied states by tying triphone states that belong to the same tied state across all PDTs in each phone state unit. Several methods are investigated to estimate the weighting parameters for model combination in each RF tied state. Furthermore, in order to reduce the acoustic score computation load and improve decoding speed, we cluster the GDFs in each RF tied state into a smaller number of classes, and use a mixture of the prototypes of the classes as the acoustic model for the RF tied state.

7.1 Random Forests-based PDTs in Acoustic Modeling

We explore the potential power of RF to generate multiple PDTs for each state of each phone unit, where phone units are described by hidden Markov models (HMM) with 3 emitting states per HMM as defined in HTK [41]. Since each PDT defines a tying structure for triphone states of a phone state, by using multiple PDTs multiple tying structures of triphone states are generated for each phone state. Through the proposed RF-based state tying across the PDTs we are able to generate more specific and yet more robust models that are unattainable from the commonly used single PDT approach.

We propose a modified question selection method to train PDTs in order to simplify the construction of random forests of speech acoustic models which are much more complex than the problems considered in [28]. We randomly (with uniform probability) choose a subset of m phonetic questions without replacement out of a total of M questions to train one set of PDTs for all phone states, with one PDT for one phone state, where the procedure of constructing a PDT is the same as the commonly adopted deterministic greedy method. We then randomly choose another subset of m questions to generate

another set of PDTs and so on. Different from the method of [28], for each PDT we use the full set of training data of each phone state instead of randomly sampling the data to avoid completely dropping out some triphone state data that may result in weak models. Completely losing some triphone data is an issue since our training data set is small, where about 33% triphone states occurred only once or twice, and about 51% triphone states occurred no more than five times.

Multiple sets of PDTs are thus generated for all phone-state units, and from which we obtain multiple sets of acoustic models MS_1,…, MS_K. In contrast, in the conventional PDT approach, there is only one set of PDTs and accordingly one set of acoustic models. Each set k has N_k models $M_{1k},\cdots,M_{N_k k}$ corresponding to the N_k tied states in the kth set of PDTs. Each triphone state S_r therefore has K models $M_{r_1 1},\cdots M_{r_K K}$, where r_k is a mapping from the triphone state S_r to a tied state in the model set k. If for two triphone states S_i and S_j, $i_k = j_k$ for $k = 1,\cdots K$, that is, S_i and S_j belong to the same tied state in each of the K PDTs, then we say that S_i and S_j belong to the same RF tied state. Fig. 7.1 illustrates the construction of RF tied states from two PDTs.

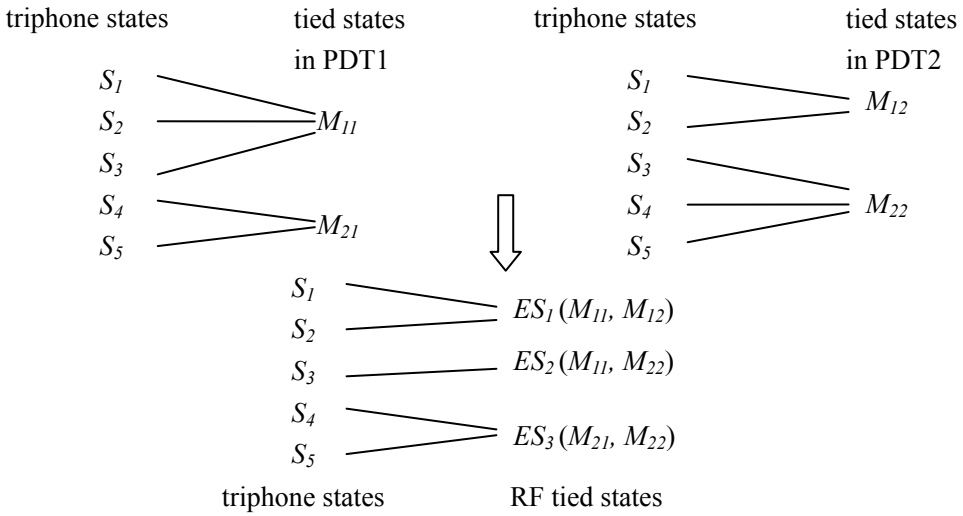
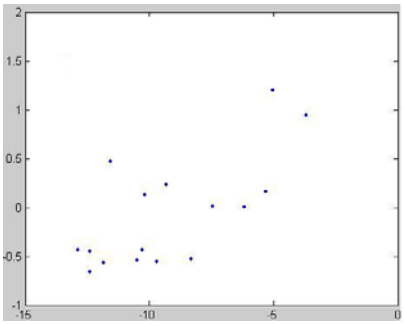


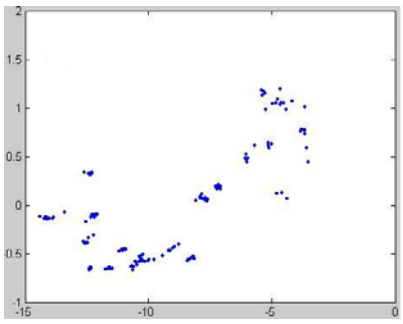
Fig. 7.1 An illustration of RF tied states generated from two PDTs

The triphone states of a RF tied state ES_l share the same multiple models $M_{l_1 1},\cdots,M_{l_K K}$.

In conventional single PDT-based method, a triphone state corresponds to a tied state which is modeled by a Gaussian mixture density (GMD). In the proposed RF-based method, a triphone state corresponds to a RF tied state which is modeled by multiple GMDs. To compare the effects on triphone modeling by a single GMD and by multiple GMDs, in Fig. 7.2, we show the Gaussian component densities for the 3rd state of triphone iy-dh+d as generated by the two methods, where each GMD has 16 Gaussian component densities, and the RF was generated by K=10, M=216, and m=200. The Gaussian densities are plotted against the first two principle components derived from 39 speech feature dimensions of training speech data, where horizontal axis is the first principle component and vertical axis is the second one. In part (a) we plot the means of different GDFs, and in part (b) we plot the one-standard-deviation contours of the GDFs. Compared with the single-PDT model, the multiple-PDT model provides a denser and broader coverage of the feature space.

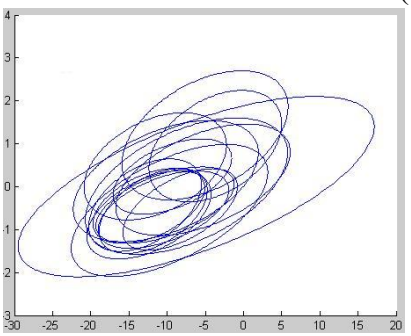


Single PDT

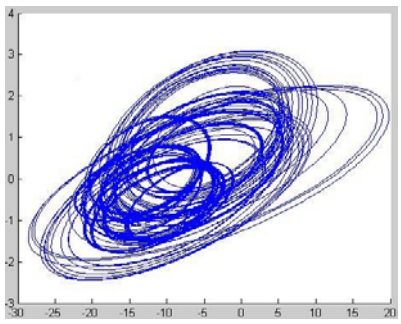


Multiple PDTs

(a) Mean points.



Single PDT (16 GDFs)



Multiple PDTs (16 GDFs/GMD, 10 PDTs)

(b) One-standard-deviation contours.

Fig. 7.2 Mixtures of Gaussians obtained by single PDT and multiple PDTs for the *3*rd state of triphone iy-dh+d.

Since there are multiple models for each RF tied state, how to effectively use these models needs to be investigated. Considering the notion that the power of multiple classifiers over a single one lies in the complementary properties of individual classifiers, and acoustic models obtained from different PDTs in RF are potentially complementary due to diversified tying structures of triphone states, we propose to combine acoustic models of each RF tied state to generate more accurate acoustic scores for each speech feature vector, and thereby improve word recognition accuracy. In [57], the complementary properties of different acoustic models were exploited by combining multiple hypothesis outputs at the word level.

7.2 Combination of Acoustic Scores

As discussed above, in speech decoding search we need to combine the multiple models in RF tied state ES_l to compute an acoustic score for each speech feature vector $\underline{x}_t$, that is,

$$P(\underline{x}_t \mid ES_l) = F(\underline{x}_t, M_{l_1 1,} \cdots, M_{l_K K}) \tag{7.1}$$

We consider linearly combining acoustic scores obtained from each acoustic model, giving

$$P(\underline{x}_t \mid ES_l) = \sum_{k=1}^{K} w_{tlk}\, p(\underline{x}_t \mid M_{l_k k}) \tag{7.2}$$

where $p(\underline{x}_t \mid M_{l_k k})$ is a Gaussian mixture density score. We need to estimate the weights w_{tlk}, which may vary with feature x_t or model $M_{l_k k}$, with $\sum_{k=1}^{K} w_{tlk} = 1$ and $w_{tlk} \geq 0$.

Following the majority vote idea of RF, the weights w_{tlk} can be set uniformly to yield an average score

$$P(\underline{x}_t \mid ES_l) = \frac{1}{K} \sum_{k=1}^{K} p(\underline{x}_t \mid M_{l_k k}) \tag{7.3}$$

On the other hand, since each PDT is trained using a set of randomly selected phonetic

questions, and different questions may have different effects on state tying in different phone states, the weights w_{tlk} can be set to be dependent on different question sets k as well as on phone states. For example, in a five-state HMM, phonetic questions concerning a left context may be more significant on the 2nd state, while the questions concerning a right context may be more effective on the 4th state. In another view, since different Gaussian density functions cover different regions in the feature space, the weights w_{tlk} should be dependent on different features $\underline{x}_t$. Therefore, assigning weights uniformly for different models may be too simplistic. In the following we propose three methods to estimate the weights.

7.2.1 Maximum Likelihood Based Weight Estimation

First, we consider weights that are specific to each RF tied state, that is

$$P(\underline{x}_t \mid ES_l) = \sum_{k=1}^{K} w_{lk}\, p(\underline{x}_t \mid M_{l_k k}) \tag{7.4}$$

Let $X = (\underline{x}_1, \cdots, \underline{x}_T)$ be i.i.d observations drawn from a RF tied state ES_l. The likelihood function is

$$L(X \mid \underline{w}_l) = \prod_{t=1}^{T} (\sum_{k=1}^{K} w_{lk}\, p(\underline{x}_t \mid M_{l_k k})) \tag{7.5}$$

Denote $\underline{w}_l = (w_{l1}, \cdots w_{lK})$. We have maximum likelihood estimation (MLE) of $\underline{w}_l$ as the following:

$$\hat{\underline{w}}_l = \arg\max_{\underline{w}_l} \{\prod_{t=1}^{T} (\sum_{k=1}^{K} w_{lk}\, p(\underline{x}_t \mid M_{l_k k}))\} = \arg\max_{\underline{w}_l} \{\sum_{t=1}^{T} \log(\sum_{k=1}^{K} w_{lk}\, p(\underline{x}_t \mid M_{l_k k}))\} \tag{7.6}$$

where the maximization is subject to the constraint of $\sum_{k=1}^{K} w_{lk} = 1$ and $w_{lk} \geq 0$.

Since there is no analytical solution for formula (7.6), we use EM algorithm [54] to iteratively compute $\underline{w}_l$. The reestimation formula is straightforwardly derived as

$$w_{lk}^{r+1} = \frac{1}{T}\sum_{t=1}^{T}\frac{w_{lk}^{r}\, p(\underline{x}_t \mid M_{l_k k})}{\sum_{j=1}^{K} w_{lj}^{r}\, p(\underline{x}_t \mid M_{l_j j})} \tag{7.7}$$

where $w_{lk}^{0} = 1/K$ for $k = 1,\cdots,K$, and $r = 0,1,\cdots$.

We used baseline acoustic models as described in [19] and HTK [41] to align a speech feature sequence with its phone state sequence for each utterance in the training data set. The EM algorithm was then used to estimate the weights for each set of models in each RF tied state.

7.2.2 Confidence Score Based Weight Estimation

Here we consider using confidence scores of acoustic models to determine the time-dependent weights. The idea is that if the confidence score of a certain acoustic model $M_{l_k k}$ on the feature x_t is large, i.e., we have more confidence that x_t is generated by $M_{l_k k}$, then $M_{l_k k}$ should have contribute more to the computation of x_t's acoustic score, and vice versa. In the current work, we consider using the confidence score accumulated probability (AP) as defined in chapter 4.

Since in general different Gaussian density functions cover different regions in the sample space, the value of AP of the feature $\underline{x}_t$ on the Gaussian mixture distribution $M_{l_k k}$, $AP(\underline{x}_t \mid M_{l_k k})$, can be used to measure the fit between $\underline{x}_t$ and $M_{l_k k}$. The larger $AP(\underline{x}_t \mid M_{l_k k})$ is, the closer is $\underline{x}_t$ to $M_{l_k k}$. We set the weight of $M_{l_k k}$ for $\underline{x}_t$ w_{tlk} to be directly proportional to the P-value $AP(\underline{x}_t \mid M_{l_k k})$, that is

$$w_{tlk} = \frac{AP(\underline{x}_t \mid M_{l_k k})}{\sum_{j=1}^{K} AP(\underline{x}_t \mid M_{l_j j})} \tag{7.8}$$

7.2.3 Relative Entropy Based Weight Estimation

We next consider using relative entropy (R-entropy) measured in a set of acoustic models

as the weight of the set of acoustic models, upon normalizing over different sets of models. For a speech feature vector $\underline{x}_t$, the acoustic score $p(\underline{x}_t \mid M_{l_k k})$ measures the likelihood of $\underline{x}_t$ being emitted by the model $M_{l_k k}$. If the values of $p(\underline{x}_t \mid M_{1k}), \cdots, p(\underline{x}_t \mid M_{N_k k})$ are spread out, then the *k*th set of acoustic models is discriminative and therefore should have a large weight, in comparison with other sets of acoustic models. Relative entropy can be used to measure the distribution spread of acoustic scores within each set of acoustic models, which is defined as the Kullback–Leibler divergence (KLD) from the distribution of acoustic scores of the models in the *k*th set of acoustic models to the uniform distribution $1/N_k$, that is

$$D_{tk} = \sum_{j=1}^{N_k} p_{tjk} \log \frac{p_{tjk}}{1/N_k} = \sum_{j=1}^{N_k} p_{tjk} \log p_{tjk} + \log N_k \tag{7.9}$$

where the distribution of acoustic scores is defined by $p_{tjk} = \dfrac{p(\underline{x}_t \mid M_{jk})}{\sum_{n=1}^{N_k} p(\underline{x}_t \mid M_{nk})}$. A large value of D_{tk} means that the scores from the *k*th set of acoustic models deviate significantly from the uniform distribution of $1/N_k$. So we define the weight w_{tlk} as

$$w_{tk} = \frac{D_{tk}}{\sum_{n=1}^{K} D_{tn}}, \text{ for } k = 1,2,\cdots K \tag{7.10}$$

Note that different from the MLE and confidence score methods, the relative entropy based weights do not vary across RF tied states.

In addition to the three methods presented above, the acoustic scores may also be combined in a time-dependent way by using the maximum score, or average *n*-best scores of multiple models. As mentioned above, the weights may also be set uniformly. In the experiments, we also evaluated these methods along with the proposed MLE, confidence score and relative entropy methods and compared the results.

7.3 **Clustering Gaussian Density Function on RF Tied States**

With the multiple sets of acoustic models generated by RFs, decoding search becomes slow since many more acoustic scores need to be computed for each input speech frame. If we can speed up the computation of acoustic scores, the total decoding time will be reduced. One possible way of reducing the load of acoustic score computation is using subspace distribution clustering HMM (SDCHMM), where in each feature dimension the Gaussian distributions of all phone states are approximated by a number of distribution prototypes. Compared with directly computing the acoustic scores from the RF based models, SDCHMM improved the decoding speed by about 20%. The results are provided in Table 8.18. However, the 20% improvement is insufficient when the forest size is large. Below we describe using model clustering to more effectively reduce the computation time.

As discussed above, the acoustic score of feature $\underline{x}_t$ for a RF tied state l is $P(\underline{x}_t \mid ES_l) = \sum_{k=1}^{K} w_{tlk}\, p(\underline{x}_t \mid M_{l_k k})$, and $p(\underline{x}_t \mid M_{l_k k}) = \sum_{i=1}^{I} \omega_{l_k ki}\, p(\underline{x}_t \mid G_{l_k ki})$, where in the model $M_{l_k k}$, $G_{l_k ki}$ is the ith GDF and I is the number of GDFs. By combining the two expressions we obtain

$$P(\underline{x}_t \mid ES_l) = \sum_{k=1}^{K} \sum_{i=1}^{I} w_{tlk}\, \omega_{l_k ki}\, p(\underline{x}_t \mid G_{l_k ki}) \tag{7.11}$$

Therefore $P(\underline{x}_t \mid ES_l)$ is a linear combination of the $K \times I$ GDFs $p(\underline{x}_t \mid G_{l_k ki})$. By computing the model combination weights with the MLE method described in section 7.2.1, which are independent of t, and combining the weights with the GMD mixture weights, we have

$$P(\underline{x}_t \mid ES_l) = \sum_{k=1}^{K} \sum_{i=1}^{I} w_{lk}\, \omega_{l_k ki}\, p(\underline{x}_t \mid G_{l_k ki}) = \sum_{k=1}^{K} \sum_{i=1}^{I} w'_{l_k ki}\, p(\underline{x}_t \mid G_{l_k ki}). \tag{7.12}$$

From (7.12), we can view the combined acoustic model for a RF tied state ES_l as an

expanded mixture of Gaussian density functions.

From Fig. 7.2 we observe that there exist large overlaps among the GDFs from multiple models. Therefore we may reduce redundant GDFs without sacrificing the feature space coverage. Toward this end, we present two methods to cluster GDFs in each RF tied state to obtain compact acoustic models, and we use Overlap accumulated probability (OAP) to measure similarity between two GDFs.

7.3.1 *K*-means Method

We use *K*-means algorithm to cluster the $K \times I$ GDFs into a certain number of classes and compute a prototype for each class. The compacted acoustic model becomes a mixture of the prototypes. The optimal prototype, $g(\hat{\mu}_p, \hat{\sigma}_p^2)$ with weight $\hat{w}_p$, for a class of N GDFs, $f(\mu_i, \sigma_i^2)$ with associated mixture weight w_i, $i = 1 \ldots N$, is obtained by the following formulas:

$$\hat{w}_p = \sum_{i=1}^{N} w_i \tag{7.13}$$

$$\hat{\mu}_p = \frac{1}{\hat{w}_p} \sum_{i=1}^{N} w_i \mu_i \tag{7.14}$$

$$\hat{\sigma}_p^2 = \frac{1}{\hat{w}_p} \sum_{i=1}^{N} w_i [\sigma_i^2 + (\mu_i - \hat{\mu}_p)^2] \tag{7.15}$$

To initialize *K*-means clustering for *P* classes, we randomly select *P* GDFs with one as the prototype of one class. Subsequently, the algorithm iterates between class assignment and prototype update.

7.3.2 Bottom-up method

As an alternative to *K*-means algorithm, we use a bottom-up agglomerative method to cluster GDFs. In contrast to the *K*-means algorithm, we do not need the prototype initialization in the bottom-up method. At each step, we choose two GDFs whose similarity is the highest among all the GDF pairs. We combine two GDFs by using

formulas (7.13)-(7.15) with N replaced by 2. The procedure is carried out iteratively until the number of prototypes equals a predefined number P. Although the number of prototypes can potentially be determined by setting a similarity threshold, we found experimentally that recognition performance was degraded if the numbers of GDFs vary significantly among the RF tied states and therefore a fixed number P is chosen for all RF tied states.

Chapter 8

Experiments and Analysis

8.1 Experimental Setup of Telemedicine Automatic Captioning System

All the proposed methods except the fast confusion network algorithm have been evaluated on the Telemedicine automatic captioning system developed in the Spoken Language and Information Processing Laboratory at the university of Missouri-Columbia. The objective of this project is to develop an online captioning system to help patients with hearing impairments communicate with doctors in teleconferencing. Developing such a system is challenging in several ways. First, Telemedicine conversations are spontaneous and contain various amounts of filled-pauses, repetitions, repairs and noises. Second, relatively sparse training data make it difficult to train a large number of parameters in both acoustic and language modeling. Third, variations in speaking style and fluency call for effective methods to describe the pronunciation patterns of different speakers. Besides development and evaluation of new algorithms, acoustic modeling efforts also include tedious work on data collection and preprocessing, feature analysis as well as noise and filled pause modeling. For a detailed description of this project, please refer to [19]. The overall structure of the Telemedicine automatic captioning system is shown in Fig. 8.1.

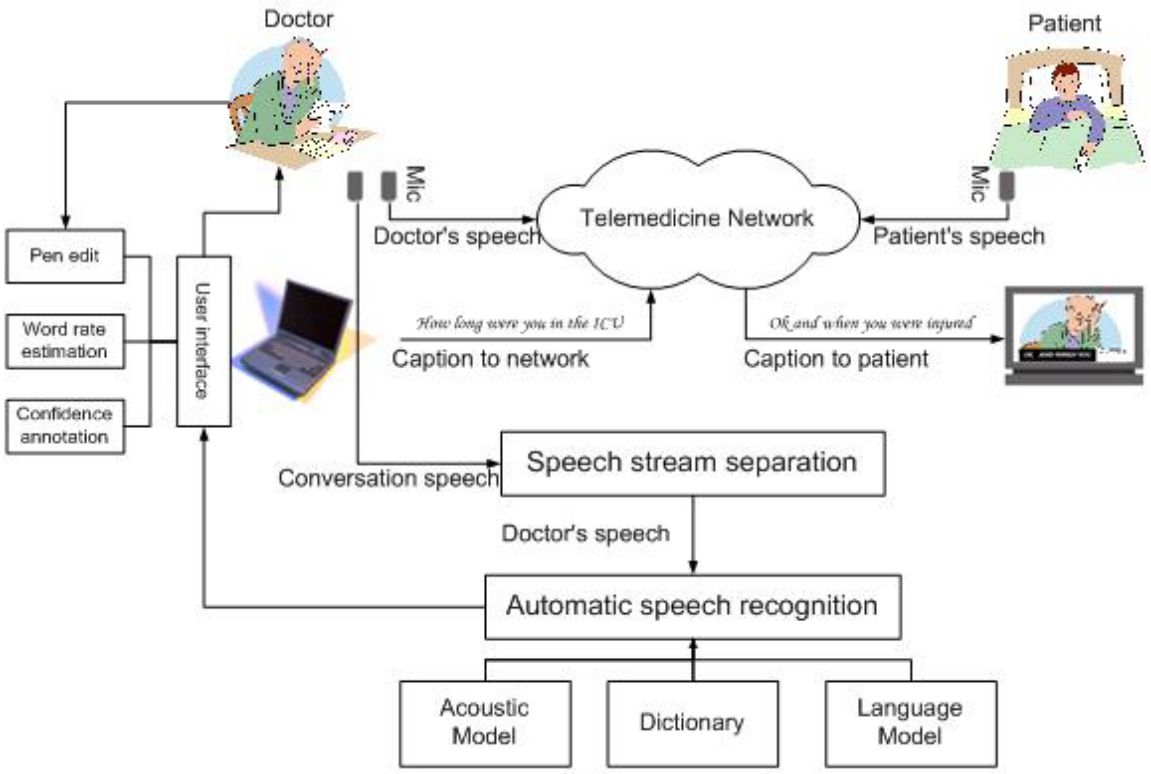


Fig. 8.1 Automatic captioning system for Telemedicine

8.1.1 Data Collection

The University of Missouri Telemedicine Network was used as the site for data collection. Recordings were made in sessions, with one session for one client, and each session lasted about 20~30 minutes. Conversation topics were primarily on neuropsychology, internal medicine, and dermatology. Health care providers' speech data were automatically extracted from conversation speech and organized into records that were separated by pauses (see Section 8.1.3).

Wireless microphone system was chosen for speech recording, where a lapel microphone was pinned to cloths, which was unobtrusive to lip reading by hearing impaired clients, non-tethering to provider, and yielded SNR of 25 dB that was higher and more consistent than SNR of far field microphone used in Telehealth. Sampling rate of 16 KHz was used in recording.

8.1.2 Preprocessing of Speech Data

The original speech recordings and word transcriptions were cleaned, aligned, and organized into a Telemedicine corpus. Currently, speech recording data of five speakers have been processed. A summary of the Telemedicine corpus is given in Table 8.1. The conversation set consist patients' speech, and the training and test data sets which altogether constitute doctors' speech. Word counts from transcription texts are also listed.

Table 8.1 Datasets used: speech (min.)/text (no. of words).

	Training set	Test set
Dr. 1	210/35,348	29.8/5085
Dr. 2	200/39,398	14.3/2759
Dr. 3	145/28,700	19.3/3248
Dr. 4	180/39,148	27.8/6421
Dr. 5	250/44,967	12.1/3988
Total	985/187,561	103.3/21501

8.1.3 Speech Stream Separation

Two methods were investigated for separating speech streams of health care provider and client. One is based on statistical speaker identification with online adaptation of speaker models [59], and the other is based on echo-cancellation capability of teleconferencing, where the former uses single channel recording and the latter uses dual channels. The echo-cancellation based method is adopted in the currently described system due to its higher reliability, even though average error rates of the two methods are comparable. The dual channel recording works in the following way. On doctor's site, wireless microphone acquires one channel input of speech conversation z, where provider's speech is directly recorded and client's speech come from a loud speaker mounted on the wall in provider's room. The second channel input is from teleconferencing audio output of patient's speech y. Sliding window of 0.1 sec. is applied to both recording channels in synchrony. For a window W(t), energy levels in the two channels, Ez(t) and Ey(t), are

compared against respective thresholds, Tz and Ty, and a label L(t) is assigned to the current window data by the following classification rule: if Ey(t) > Ty, then L(t) = patient; else if Ez(t) > Tz, then L(t) = doctor; else, L(t) = pause. Note that the decision rule discards competing speech of provider and client.

A counter is used to track successively detected pauses. If the count exceeds a threshold Tc, then the current record of doctor's speech is ended and a new record is created. Speech records thus created captures from half a sentence up to five sentences, with majority of records containing one or two sentences. Subsequent speech acoustic model training and system evaluation tests are performed on speech records.

8.1.4 Acoustic Model

Speech feature consisted of 39 components: 13 MFCC parameters $c_0, \cdots c_{12}$ and their 1st and 2nd order time derivatives. Short-time analysis window size was 20 ms and shift was 10 ms. A total of 52 acoustic sound units were defined, including 42 speech monophone units, seven filled pause units, one unit for sound artifacts like lip smack and microphone ruffling, as well as one pause and one silence unit. Context-dependent triphone modeling was used for speech, and context independent modeling was used for the rest sound units. Gaussian mixture density based Hidden Markov model with three emitting states was used for triphone and other sound units, where each emitting state was modeled by a size-16 Gaussian mixture density with diagonal covariance matrix. Baseline acoustic model parameters were estimated by using the HTK toolkit [19] where HMM states were tied by phonetic decision trees.

8.1.5 Language Model

The described captioning task involves spontaneous dialog style speech in various medical specialty domains. The major difficulty faced by the language modeling task here is that transcriptions of Telehealth conversation were very limited, and there were little accessible textual corpora elsewhere that match the domain and style of Telehealth.

In order to improve quality of trained language model (LM), textual data from other domains were used to enlarge vocabulary coverage and improve events estimation, including public domain data sets of Switchboard, Broadcast News, and Call Home, and medical domain data sets of medical written report for good coverage of medical vocabulary, and an additional Telehealth related dataset on dermatology. The datasets were categorized as in-domain and out-of-domain, where the in-domain ones included Telehealth transcription sets and the Telehealth related dermatology set, and the out-of-domain ones included Switchboard, Broadcast News, Call Home, and medical written reports.

Trigram LM was trained with Kneser-Ney backoff [60] by using the SRI toolkit [61]. Two class trigram LMs were trained for the in-domain datasets, and four word trigrams LMs were trained for the out-of-domain datasets. The six LMs were linearly interpolated by using a ten-fold validation on Telehealth training set. The interpolation weights were optimized by a greedy forward selection algorithm [62]. The in-domain class trigram and out-of-domain word trigram combination, referred to as ICOW LM, was used in all of the experiments on Telehealth captioning for this dissertation work. Details of language modeling for Telehealth captioning is described in [62].

8.1.6 Decoding Engine

The speech decoding engine, TigerEngine v1.1, was developed in the Spoken Language and Information Processing Laboratory, Department of Computer Science of University of Missouri-Columbia, USA [32]. The decoding system performs large vocabulary continuous speech recognition in real-time based on one-pass time-synchronous Viterbi beam search, and its search organization is based on the lexical tree-copy algorithm [15]. Acoustic and language knowledge sources, including cross-word triphone HMMs and trigram LM are integrated as early as possible in search organization with a trigram LM lookahead.

An novel feature of TigerEngine is a very fast and memory efficient language model

lookup method for trigram-based language model lookahead, called Order-Preserving LM Context Pre-computing (OPCP). OPCP reduced LM lookup time to about 10% total decoding time without decrease of word accuracy. The total memory cost of OPCP for LM lookup and storage was about the same or less than the original N-gram LM storage. Both the percentage of decoding time and the memory usage of OPCP are much less than commonly used methods in comparable decoders, and the time and memory advantages of OPCP become more pronounced with the increase of LM size. Details of the speech decoding engine is described in [32].

8.2 Experiments for Confusion Network

Experiments of confusion network were conducted on the Switchboard 2001 HUB- 5 Corpus. HTK toolkit was employed to generate 310 word lattices for a test set of 310 sentences, by using triphone acoustic models and bigram language models. The acoustic models were provided by ISIP and language models by SRI.

A comparison of execution time of word hypothesis sequence generation between the proposed CN algorithm and MBS-CN is shown in Table 8.1. The evaluation result of Table 8.2 was obtained by running the three algorithms on a 2GHz Pentium-4 processor and the results were averaged over the test set. Because MBS-CN was extremely slow, we pruned 95% of the links in MBS-CN, but pruned none in proposed CN.

Table 8.2 Comparison of execute time for CN algorithms with different lattice sizes

Method	Execution Time (× real time)					
	Average number of links/Lattice					
	0-1k	1k-2k	2k-4k	4k-6k	6k-8k	8k-10k
MBS-CN	< 0.01	0.05	0.1	0.15	0.4	0.6
Proposed-CN	< 0.01	0.06	0.07	0.08	0.08	0.09

From Table 8.2 we observe that when the number of links per lattice is small, two

algorithms both run very fast. With the lattice growing larger, especially when number of links was larger than 6k, the MBS-CN algorithm required more than 0.4*real time.

Recognition word error rates were further compared across the two algorithms and the results are shown in Table 8.3.

Table 8.3 Comparison of decoding results for CN algorithms

Method	Word Error Rate (%)
HTK's one-best	47.57
MBS-CN	47.17
Proposed-CN	47.01

The results indicate that the expected word error minimization indeed yield lower word error rate than sentence error minimization (the case of HTK's one-best).

8.3 Experiments for Confidence Annotation

Experiments were conducted on the Telemedicine automatic captioning system, using five doctor's data. The kernel function we used for SVM is radial function $K(x_i, x_j) = \exp(-\gamma | x_i - x_j |^2)$, where γ was set to be 1/(data dimension) and slack variable ξ was 0.1. The value of *m* for Random Forests was 4 , the value of *N* was 0.632 time of sample size, and the number of trees was 500. We used software R [29] to train the models.

To avoid over-fitting, we used 10-fold cross-validation for each technique, i.e., divided the dataset randomly into 10 equal-sized subsets. We kept one of the 10 subsets as the validation set, and combined the remaining 9 subsets to form the training set. Repeating this 10 times, the accuracy measure was averaged on the 10 validation sets. The annotation error rate and word error rate are summarized in Table 8.4, where

$$Annotation \quad Error \quad Rate = \frac{Number \quad of \quad incorrect \quad annotations}{Total \quad number \quad of \quad annotations} \tag{8.1}$$

$$Word\quad Error\quad Rate = \frac{Substitution + Insertions}{Total\quad number\quad of\quad words\quad in\quad hypotheses} \tag{8.2}$$

Table 8.4 Performance of three classifiers

	Word Error Rate	Annotation Error Rate		
		Decision Tree	SVM	RF
Dr. 1	15.54%	13.54%	13.50%	13.16%
Dr. 2	19.01%	18.22%	18.52%	17.55%
Dr. 3	21.46%	18.34%	18.34%	17.83%
Dr. 4	18.35%	16.05%	16.41%	15.90%
Dr. 5	15.50%	13.90%	13.40%	13.26%

It is clear that Random Forests achieved best results for each doctor, where the false alarm rate was from 3.58% to 4.76%.

Fig. 8.2 shows the importance of different features ranked by Random Forest, averaged on different data sets. From left to right the features are: Acous-Score, Langu-Score, Langu-Type, Total-Score, AP, Ave-AP, LWPP, Ave-LWPP, Entropy, Pos-Score, Bi-Pos and Tri-Pos.

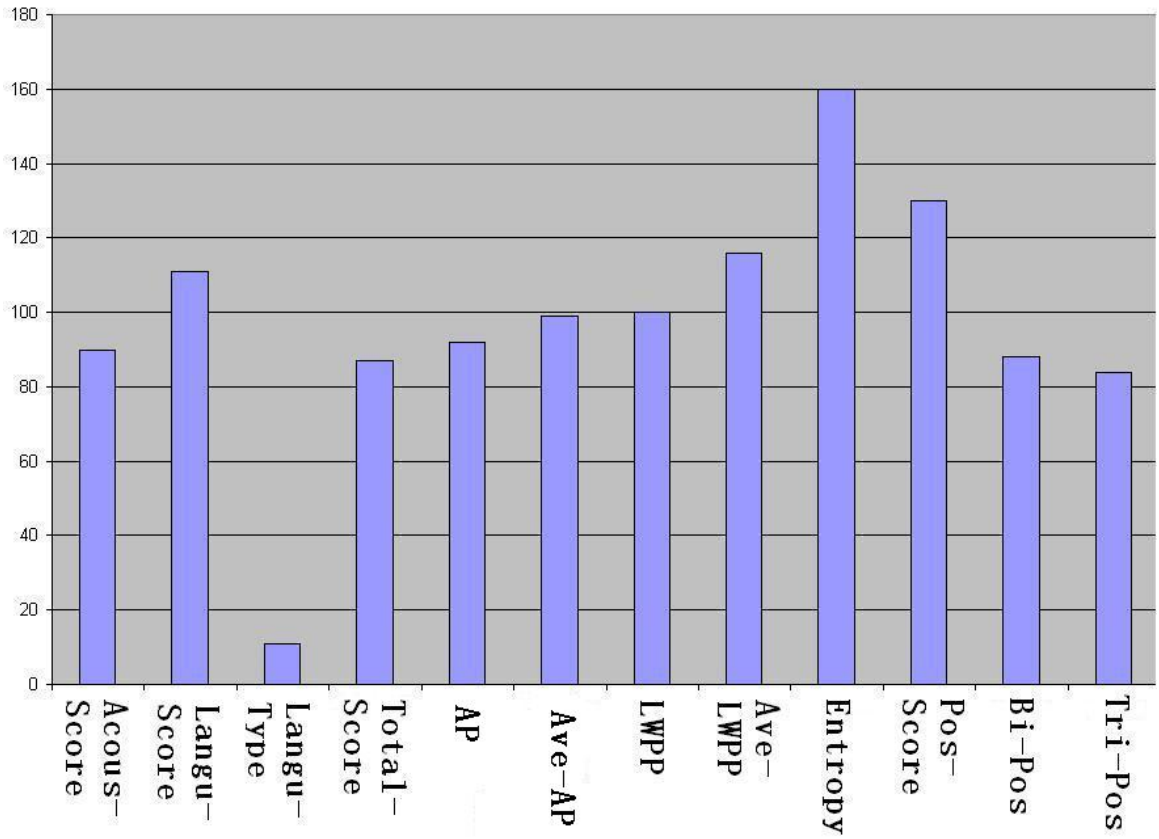


Fig. 8.2 Histogram of importance of features

From Fig. 8.2 we know that Entropy is most important, since it measures the difference of word posterior probabilities between the output word and the other words. Langu-Type almost has no effect, we can delete it in the confidence annotation.

To further measure the importance of the proposed features, we also get the performance of confidence annotation on Dr. 1's dataset using different combination of features by using Random Forests. Starting from using all 12 features, individual features were selectively removed to see its effect on annotation error rate. The results are summarized in Table 8.5.

Table 8.5 Performance of confidence annotation using different combination of features

	Annotation Error
Using all features	13.16%

No Entropy	13.50%
No Pos_Score	13.20%
No Entropy and Pos-Score	14.14%
No AP and Ave-AP	13.40
No Bi-Pos and Tri-Pos	13.34
No Entropy, AP, Ave-AP, Bi-Pos and Tri-Pos	14.16

From Table 8.5 we observe that when Entropy was not used, error rate increased by 0.34%. Since Entropy and Pos-Score are correlated to some extent, when we delete them both, the performance decreased by 1.02%. From Table 8.5 we also observe that when AP and Ave-AP were not used error rate increased by 0.24%, and when Bi-Pos and Tri-Pos were not used, error rate increased by 0.18%. So AP, Bi-Pos and Tri-Pos all played important roles in confidence annotation.

To further understand the proposed features for confidence annotation, the distributions of Entropy for “incorrect” class and “correct” class are shown in Fig. 8.3. In general, Entropy of words in “correct” class are smaller than those in “incorrect” class, and therefore Entropy is an important feature in confidence annotation.

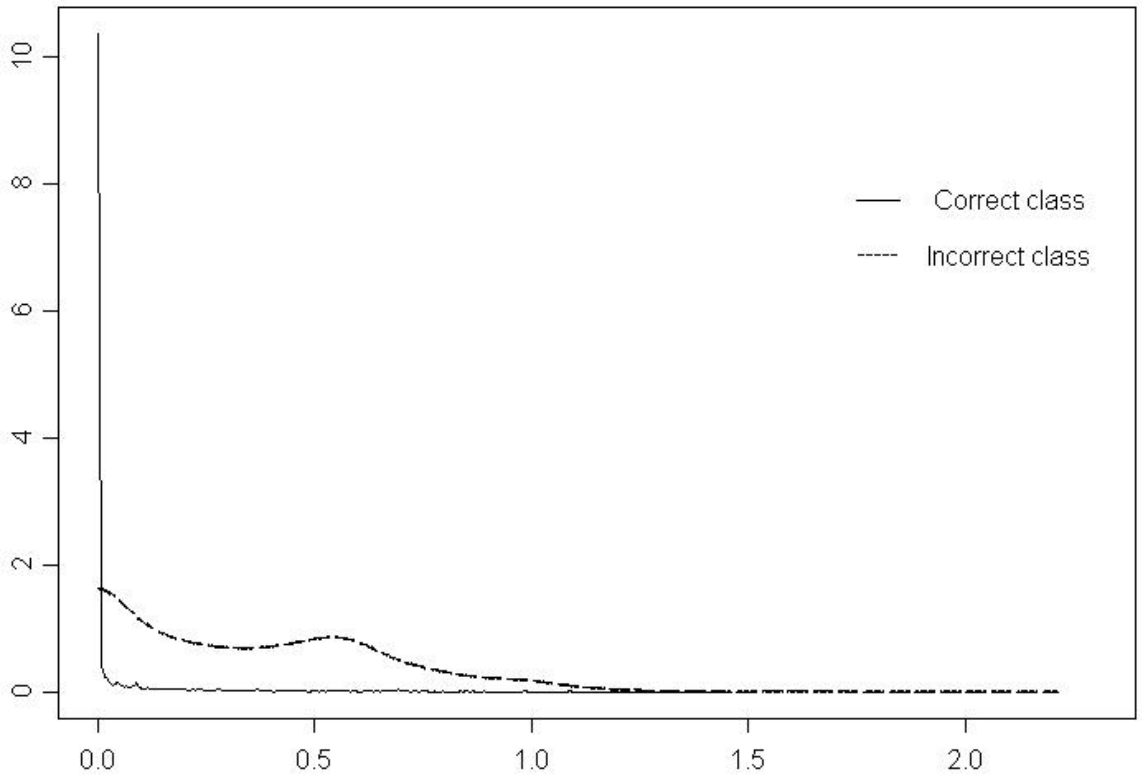


Fig. 8.3 Distribution of Entropy in both “correct” and “incorrect” classes.

Table 8.6 summarized the results of Random Forests on Dr. 1’s dataset when training different number of trees. From Table 8.6 we observe that as the number of trees increased, the error rate decreased to some extent.

Table 8.6 Performance of confidence annotation using different numbers of trees

No. of trees	Annotation Error
10	14.60%
20	14.14%
50	13.38%
100	13.20%
200	13.18%
500	13.16%
700	13.16%
800	13.18%

8.4 Experiments for New Improvements in Decoding Engine

Experiments were conducted on the Telemedicine automatic captioning system to evaluate the improvement techniques described in chapter 5.

Table 8.7 shows the recognition results for SDCHMM, using the three methods discussed in chapter 5 to measure similarities of *pdf*'s.

Table 8.7 Comparison of word accuracy between SDCHMM and CDHMM

	Word Accuracy			
	Full-space model	Subspace model		
		CBB	KLD	OAP
Dr. 1	80.28%	80.41%	80.50%	81.14%
Dr. 2	74.08%	73.90%	73.76%	74.05%
Dr. 3	71.82%	73.98%	73.55%	74.41%
Dr. 4	79.32%	79.66%	79.13%	79.54%
Dr. 5	82.12%	81.42%	81.85%	82.17%
Average	77.52%	77.87%	77.76%	78.26%

From Table 8.7 we see that OAP obtained better results than CBB and KLD, except for Dr. 4's dataset, where CBB got the best result. We also observe that the average performance of each subspace model is better than the original full-space model, which is mainly due to our insufficient training data in training a large set of CDHMMs. Subspace modeling made up for the deficiency of training data by reducing the large set of parameters of CDHMMs to a smaller set of parameters of SDCHMM.

In order to measure the effects of Confidence-based pruning (CBP) and SDCHMM, In Table 8.8 we summarize the word recognition accuracy results and decoding speeds on Dr. 1's dataset, before and after using Confidence-based pruning and SDCHMM. Here SDCHMM used OAP, and the baseline used CDHMMs. The thresholds for confidence-based pruning were empirically set to be -80 for $AP(w)/T$ and -9 for $LWPP(w)/T$, with T the duration of word w.

Table 8.8 Performance of confidence-based pruning and SDCHMM

		Word accuracy	Speed (RT factor)
Baseline		79.45%	1.2
CBP(AP + LWPP)		79.35%	1.4
SDCHMM		80.29%	1.0
CBP+ SDCHMM	AP only	80.25%	0.9
	LWPP only	80.27%	0.9
	Both	80.25%	0.8

From Table 8.8 we observe that by using Confidence-based pruning and SDCHMM, the speed of decoding engine is improved from 1.2 real-time to 0.8 real-time. But when only confidence-based pruning was used, the speed was even slower than baseline, since computing AP took up more time than that saved by pruning of uncompetitive paths. Furthermore, in confidence-based pruning, when only AP or LWPP was used, the speeds were both about 0.9 real-time. Therefore it is meaningful to use both as the confidence scores for path pruning.

Table 8.9 compares recognition results obtained by using the proposed pre-backtrace method and the baseline, where the latter used OAP based SDCHMM. The pre-backtrace duration thresholds based on unfilled pause, pitch, and filled pauses were empirically set to be 200 ms , 100 ms, and 50 ms, respectively.

Table 8.9 Recognition performance of baseline and pre-backtrace

	Word Accuracy	
	Baseline	Pre-backtrace
Dr. 1	81.14%	78.61%
Dr. 2	74.05%	73.32%
Dr. 3	74.41%	72.29%
Dr. 4	79.54%	79.32%
Dr. 5	82.17%	79.76%
Average	78.26%	76.66%

We observe that pre-backtrace decreased word accuracy somewhat, since it is possible to prune away the best path prematurely. Fig. 8.4 shows the distributions of latencies without and with pre-backtrace. From Fig. 8.4 we see that pre-backtrace decreased latency significantly. Without pre-backtrace, the maximal latencies ranged from 8.4 seconds to 38.6 seconds, depending on the speaking style of different doctors, and the average latencies were from 2.4 seconds to 5.7 seconds. After pre-backtrace, the average latencies were from 2.1 seconds to 4.5 seconds, and the maximal latency dropped to 12.9 seconds.

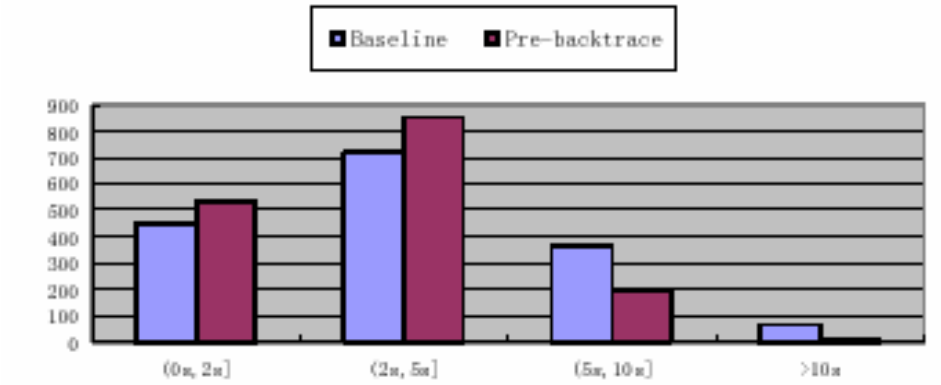


Fig. 8.4 Distributions of latencies

In order to verify the effect of using F_0 contour in pre-backtrace, we provide the results of using only unfilled pauses or filled pauses to detect boundaries for pre-backtrace. This test was done on Dr. 1's dataset. Table 8.10 summarizes the results.

Table 8.10 Comparison of using unfilled pause , filled pause and F_0 contour in detection of prosodic boundary

	Word Accuracy	Latency	
		Average	Max.
200 ms pause	79.24%	3.5s	9.6s
150 ms pause	76.23%	2.7s	7.5s
Filled pause	80.55%	3.8s	9.2s
200 ms pause + filled	78.61%	3.1s	7.7s

pause + F_0 contour			
Baseline	81.14%	4.0s	11.0s

We observe that unfilled pause feature is effective on latency but introduces a large loss on word accuracy, filled pause feature introduces less errors but has less effect on latency, and the best tradeoff in accuracy and latency appears to be achieved with using all three prosodic features.

8.5 Experiments for Novel Lookahead Phonetic Decision Tree

Table 8.11 gives the recognition accuracies and table 8.12 gives the model sizes of 5 different PDT training methods:

- Baseline: Baseline method.
- B_PSDT: Baseline plus phone-state dependent threshold.
- CLA-1: *1*-step constrained lookahead method.
- CLA-2: *2*-step constrained lookahead method.
- SFLA: stochastic full lookahead method.

We set r =20 (subtree ensemble) for stochastic full lookahead method, and n=20 (n-best) for both constrained and stochastic lookahead methods. The value C in phone-state dependent threshold is set such that the average threshold of different PDTs equals the one used in baseline method, which is 400 in our experiments.

Table 8.11 Recognition accuracies of different PDTsmethods (%)

	Dr. 1	Dr. 2	Dr. 3	Dr. 4	Dr. 5	Avg.
Baseline	81.40	74.16	76.29	78.31	82.40	78.96
B-PSDT	81.39	73.65	76.29	78.32	82.97	79.00
CLA-1	81.44	72.45	76.76	78.31	81.80	78.71
CLA-2	81.46	72.42	76.95	78.20	81.95	78.74
SFLA	81.14	73.90	76.35	78.45	82.92	79.01

Table 8.12 Model sizes of different PDTs methods
(number of tied states or acoustic models)

	Dr. 1	Dr. 2	Dr. 3	Dr. 4	Dr. 5	Avg.
Baseline	2070	1480	1093	1415	1735	1559
B-PSDT	1603	1425	908	1117	1440	1299
CLA-1	1611	1484	918	1131	1476	1324
CLA-2	1625	1509	931	1134	1501	1340
Sfla	1431	1274	879	973	1237	1159

From Table 8.11 and Table 8.12 we observe that stochastic full lookahead method reduces model size significantly (26% relative to baseline). Constrained lookahead methods do not produce consistent improvement in recognition accuracy.

Table 8.13 summarizes recognition word accuracy after combining hypotheses from different models, where n-best is based on ranking the five methods by their accuracy performance. From Table 8.13 we observe that the CN-based hypotheses combination produces a consistent improvement in recognition accuracy. Integrating more hypotheses in general yielded higher accuracy.

Table 8.13 Recognition accuracy (%) after combination with CN

	Word accuracy after combining *n*-best results			
	n=2	*n*=3	*n*=4	*n*=5
Dr. 1	81.67	81.65	81.97	81.81
Dr. 2	74.52	74.52	74.48	74.45
Dr. 3	76.91	76.88	76.94	77.03
Dr. 4	78.87	78.82	78.74	78.76
Dr. 5	83.20	83.45	83.50	83.55
Avg.	79.48	79.50	79.56	79.56

To investigate the effect of the repeat number r in generating random subtrees on performance of stochastic full lookahead, we trained PDTs with different values of r on Dr. 3's dataset and compared the model sizes and recognition accuracies. From Table 8.14 we do not see a consistent tendency.

Table 8.14 Performance of stochastic full lookahead with different *r*

r	5	10	20	50
Model size	877	861	879	855
Accuracy (%)	76.60	76.63	76.35	76.57

We also compared the performance of constrained lookahead and traditional lookahead (TLA) methods on Dr. 3 dataset. Table 8.15 gives the results. We can see that the constrained methods give better performance than the unconstrained one both in recognition accuracy and in model size.

Table 8.15 Comparison of constrained lookahead and traditional lookahead method

	Accuracy (%)	Model size
TLA-1	76.51	1064
TLA-2	76.08	1082
CLA-1	76.76	918
CLA-2	76.95	931

8.6 Experiments for Random Forests Phonetic Decision Tree

In this section we present the experimental results of Random Forests PDT. Experiments were also conducted on Telemedicine automatic captioning system.

8.6.1 RF-based PDTs with Different Methods for Model Weights

We trained multiple sets of PDTs by using the proposed RF method and obtained multiple sets of acoustic models. By default, the number of phonetic questions used for constructing a PDT *m* was set to be 200 (see discussion in Section 8.7), where the total number of phonetic questions *M* was 216 as defined in HTK. Within a set of PDTs, the same *m* phonetic questions were used. Across two different sets of PDTs, the phonetic questions differed by about 15 on average. The acoustic scores were combined by using

the proposed methods of MLE, *AP*, and R-entropy, as well as the methods of maximum score (MAX), *n*-best out of *K* models (n-K), and simple average (Uniform). In addition, weights from MLE and relative entropy were also averaged (MLE+R-entropy). Table 8.16 gives the performance in word recognition accuracy averaged over the 5 doctors' test sets.

Table 8.16 Word accuracies (%) averaged over five doctors by using the proposed acoustic score combining methods in RFs PDTs.

Baseline	78.96
n-best (5-20)	80.49
n-best (5-50)	80.62
n-best (10-50)	80.79
n-best (10-100)	80.31
n-best (20-100)	80.56

(a) Baseline and *n*-best methods with different *n*-*K*.

	K			
	10	20	50	100
MAX	80.35	80.41	79.95	79.70
Uniform	80.39	80.57	80.71	80.80
MLE	80.47	80.81	80.90	80.92
AP	80.43	80.69	80.85	80.90
R-entropy	80.39	80.64	80.88	80.91
MLE+R-entropy	80.39	80.72	80.95	80.96

(b) All other combining methods.

From Table 8.16 we observe that the proposed RF-based PDTs improved the word recognition accuracy by 0.74% ~ 2.00% absolute, and the effect was dependent on acoustic score combining methods and forest sizes. When the forest size was large (*K*=50, 100), using the averaged weights from MLE and relative entropy yielded best results.

Uniform weight was also a good choice, since the implementation was simple and yet the performance loss relative to the best case was less than 0.3% absolute. Although the AP method also produced good results, computing *AP* consumed much more time than the relative entropy method. As the forest size increased to 50 and 100, the performance of MAX method was decreased. This is because that the maximum score is susceptible to unreliable models, and as the number of models increases, the possibility that the score of some unreliable model turns into the maximum score becomes larger. The *n*-best methods yielded good results, but the results were inferior to MLE, *AP*, R-entropy, and the MLE+R-entropy.

In order to compare our proposed RF approach with the method of [57], we also trained 50 sets of PDTs by using the method of *n*-best random split selection, where the phonetic question used to split each node was randomly selected from the top-10 best questions of the 216 questions, and speech decoding evaluation was carried out 50 times by using each set of acoustic models. For each speech utterance, 50 1-best recognition hypotheses were thus generated with each obtained from one set of the acoustic models. We put the multiple hypotheses together into a simple word lattice and set the score uniformly for every path, and then aligned the lattice into a confusion network (CN) by using fast CN algorithm, as we did in section 6.5. In such a way, the posterior probabilities of the links in the word lattice were the same, and the posterior probability of a word in CN became the ratio of the number of paths that included the word at a certain position to the total number of paths in CN. This method is similar to the commonly used hypothesis integration method of ROVER [63]. Table 8.17 summarizes word recognition accuracies obtained by using the *n*-best random split selection generated PDTs and CN based hypothesis integration.

Table 8.17 Average word accuracy (%) by using *n*-best random split selection and CN.

	Word accuracy (%)

K=10	79.88
K=20	79.97
K=50	80.24

From Table 8.17 we observe that *n*-best random split selection with hypothesis integration at the output word level also improved accuracy performance, but the improvement is less than what are obtained by our proposed RF methods. Furthermore, combining outputs at the word level requires running decoding search *K* times, which is much slower than our methods of combining acoustic score when single processor computers are used. Based on experiments with our decoding system TigerEngine 1.0, the time spent for computing acoustic scores by using the baseline acoustic models is about 15 to 20 percent of total decoding time. So by using *K* sets of acoustic models with each set having a size comparable to that of the baseline, and with a large *K*, the decoding time is about 0.15*K* to 0.20*K* times as the baseline. Note that this estimate does not take into account of the effect of various path pruning in decoding search. For accurate acoustic models, effective path pruning can avoid evaluation of acoustic scores of many speech units and thereby save computation time on acoustic scores.

We also investigated the performance of the question selection method proposed in [28], which randomly (with uniform probability) chooses a subset of *m* phonetic questions without replacement out of a total of *M* questions at each node, and then selects the best question in the subset to split the node. We trained 50 sets of PDTs by using this method, where *m* was set to be 150, which was the best value based on our experiments (described in section 8.6.4). Table 8.18 summarizes word recognition accuracies thus obtained by using different score combination methods.

Table 8.18 Average word accuracy (%) by using random question selection at each node, with *K*=50, *m*=150

Score combination method	Uniform	MLE	*AP*	R-entropy	MLE + R-entropy
Word accuracy	80.98	81.02	80.87	81.00	81.00

Comparing Table 8.18 with Table 8.16, we observe that the word accuracies in Table 8.18 is slightly better than Table 8.16. However, the results in Table 8.18 were obtained with *m* equaling 150, while the results in Table 8.16 were obtained with *m* equaling 200. In any case, the difference between Table 8.16 and Table 8.18 is not significant, so we conclude that the effect of these two question selection methods on word recognition accuracy are similar. The ranking of the model combination methods are changed. In Table 8.16, the method MLE + R-entropy obtained best results with large forest size, and in Table 8.18, the method MLE got the best result. As shown in both tables the difference between these two methods are below 0.1 percent, so we conclude that the difference between MLE and MLE + R-entropy is small. Still, using uniform weights is a good choice for its simple implementation and little performance loss.

8.6.2 Experimental Results of Using SDCHMM and Compact Models

In this subsection we present the experimental results using SDCHMM and the compact models generated by the clustering methods discussed in section 7.3. Table 8.19 summarizes word recognition accuracy and decoding time for each method, where the decoding time is measured relative to the baseline decoding time. The MLE method was used to estimate the weights of model combination.

Table 8.19 Average word accuracy and decoding time (×baseline decoding time) by using SDCHMM and compact models with weights estimated by MLE.

		Average word accuracy (%)	Average time (times of baseline)
SDCHMM 16 prototypes / dimension	*K*=10	80.24	2.5
	K=20	80.41	4.8
	K=50	80.54	11
K-means (*K*=50)	16 classes	79.95	1.2
	32 classes	80.43	1.4
Bottom-up	16 classes	79.92	1.2

(*K*=50)	32 classes	80.43	1.4

From Table 8.19 we observe that after incorporating SDCHMM the decoding speed was improved but still very slow. A more effective speed up in decoding was achieved by decreasing the number of GDFs for each RF tied state. After using the *K*-means or bottom-up methods to compact the acoustic models, the decoding speed approached the baseline level, while the accuracy advantage of RF was largely maintained. The *K*-means and the bottom-up methods yielded similar performances. It is worth noting that when the number of classes was set as *P*=16, equivalent to having a mixture of 16 Gaussian prototypes in each RF tied state, the RF method yielded accuracy about 1% higher than baseline that used 16 Gaussians in each tied state. The increased decoding time was due to the fact that the average number of RF tied states was 7478, about five times the number of tied states in baseline. However, due to the effect of path pruning in decoding search, the time increased for computing acoustic scores was not as large, only about 20%. Note that SDCHMM was not used with the *K*-means or bottom-up methods, since for the compact models the overhead of SDCHMM offset its saving in Gaussian score computations.

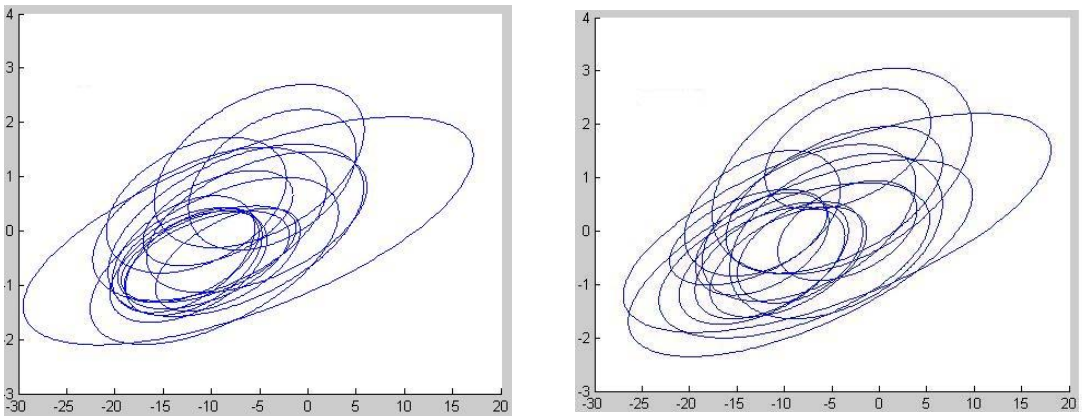
We further increased the number of classes in using the bottom-up clustering method to investigate the effect of number of prototypes on recognition performance and speed. Table 8.20 presents the average word accuracy and decoding time for seven cases of prototype numbers per RF tied state. Here we used a total of 100 sets of acoustic models, i.e., *K*=100. From Table 8.20 we observe that in general as the number of classes increases, the average word accuracy also increases, but the decoding speed decreases since the time for computing acoustic scores increases.

Table 8.20 Average word accuracy and decoding time (×baseline decoding time) of the bottom-up clustering method versus different numbers of classes per RF tied state.

Number of classes	Average word accuracy (%)	Average time (times of baseline)
16	80.00	1.2

32	80.52	1.4
64	80.60	1.9
96	80.57	2.2
128	80.75	2.4
160	80.72	2.7
192	80.83	3.0

In Fig. 8.5, we compare the Gaussian mixture densities for the 3rd state of triphone iy-dh+d from the baseline acoustic model and the compacted model obtained by *K*-means clustering on RF generated 100 GMDs. The one-standard-deviation contours are again plotted against the first two principle components derived from 39 speech feature components. We observe that although the numbers of GDFs are the same for the two models, the Gaussians in the compact model are less overlapped and cover the feature space more effectively. This is in agreement with the fact that the compacted acoustic models improved the recognition performance over the baseline.



(a) Baseline model (16 GDFs)

(b) Compacted model (16 GDF prototypes for 160 GDFs)

Fig. 8.5 One-standard-deviation contours of mixtures of Gaussians in baseline model and clustered compact model for the *3*rd state of triphone iy-dh+d.

8.6.3 Significance Test

In order to verify the effectiveness of the proposed methods, we conducted a significance

test on performance differences between the proposed methods and the baseline method.

Let m denote the number of sentences in the test data set. For the baseline acoustic models, let x_i be the word recognition accuracy of the ith sentence in the test data set and assume x_i follows Gaussian distribution with mean μ_1 and standard deviation σ_1. Also, let y_i be the word recognition accuracy of the ith sentence from using a proposed method and y_i follows Gaussian distribution with mean μ_2 and standard deviation σ_2. The null hypothesis H_0 postulated an insignificant effect of using the proposed method, i.e., $\mu_2 = \mu_1$, and the alterative hypothesis H_1 asserted a positive difference, i.e., $\mu_2 > \mu_1$. Let $t_i = y_i - x_i$ and denote the sample mean and sample variance of t_i as $\bar{t}$ and s^2. The test statistic is $T = \bar{t}/(s/\sqrt{m})$, which follows a t distribution with m-1 degrees of freedom.We reject H_0 if $T > t_{m-1,\,1-\alpha}$, where α is the error of rejecting H_0 when it is true, or the level of significance of the test.

In our test data set, m =1641 and $t_{m-1,\,1-0.001} = 3.095$. For all the cases in Tables 8.16, 8.18, 8.19, and 8.20, we obtained $T > t_{m-1,\,1-0.001}$. Therefore we conclude that our proposed methods improved word recognition accuracy significantly over baseline on the Telehealth captioning task.

We also conducted a significance test on performance differences between the proposed RF method (MLE for weights) and the method of n-best random split selection with hypothesis integration of Table 8.17. For K=10 we obtained $t_{m-1,1-0.002} < T < t_{m-1,1-0.001}$, for both K=20 and K=50 we obtained $T > t_{m-1,1-0.001}$. Therefore we also conclude that our proposed methods improved the word recognition accuracy significantly over the method of n-best random split selection with hypothesis integration.

8.6.4 Discussion on Random Forests PDT

RF tied states versus PDT tied states

In conventional single PDT state tying, each triphone state is clustered into one tied

state, and the training data of the triphone states that are tied are used to estimate the GMD parameters of one model. In the proposed RF based state tying, although each triphone state is tied into one RF tied state, the training data of each triphone state contribute to the construction of models in several RF tied states. Therefore, the data count in each RF tied state is increased, and training data in different RF tied states are partially overlapped. From Fig. 1 we can see that triphone states S_1 and S_2 are tied into RF tied state ES_1, and ES_1 has two models M_{11} and M_{12}. M_{11} is constructed using the training data of triphone states S_1, S_2 and S_3, and M_{12} is constructed using the training data of S_1 and S_2. So S_1, S_2 and S_3 contribute to the construction of the models in ES_1, with weights or repetition times of 2, 2, and 1, respectively. Similarly, S_1, S_2, S_3, S_4 and S_5 all contribute to the construction of the models in ES_2, with weights 1, 1, 2, 1, and 1, and S_3, S_4 and S_5 contribute to the construction of the models in ES_3, with weights 1, 2, and 2. In this way, the average number of triphone states contributing training data to each RF tied state is 3.67, while the average number of triphone states tied into each RF tied state is 1.67. In contrast, in PDT1 and PDT2 the average number of triphone states contributing training data to each tied state is both 2.5, which is the same as the average number of triphone states tied into each tied state. Therefore, in comparison with tied state in single PDT state tying, the data count in each RF tied state is increased, and the number of triphone states tied into each RF tied state is decreased.

In our experiment with the single PDT-based state tying to train acoustic models from the dataset of speaker Dr1, the number of tied states was 1603 and the average number of triphone states in each tied state was 9.1; but with RF based multiple PDT state tying and with K=50, the number of RF tied states was 12033 and the average number of triphone states contributing to the estimation of GMD parameters in each RF tied state was increased to 29.8. Since a RF tied state corresponds to one GMD, the RF-based acoustic model has many more GMDs than that of the baseline. This increased model resolution is possible with the RF method due to the multi-way state tying

associated with each triphone state. In contrast, the baseline model would have become very unreliable if such a large number of GMDs were used, since each GMD would have only one or two triphone states contributing data to its parameter estimation.

Recognition accuracy versus size of question subset

As mentioned before, a proper value for *m*, i.e., the subset size of split variables or phonetic questions, is needed. The default value of *m* in the R software is approximately $\sqrt{M}$ [29]. In our experiments *M* equals 216 and $\sqrt{M}$ approximately equals 15. However, we opt for setting *m* to be much larger, i.e., 200 instead of 15, based on the consideration that using too few questions in constructing a PDT will result in poor tying of triphone states and therefore poor models. Although using a very large number of acoustic models may potentially compensate for the weakness of individually poor models, the time and memory requirement of using a large number of weak models in decoding search would be too high. On the other hand, if the number of acoustic model sets (equivalent to the number of PDTs for each phone state) is limited, then each PDT should be strong, and therefore *m* should be sufficiently large. That said, we investigated the effect of different values of *m* on word recognition accuracy for a fixed forest size *K*=50. Table 8.21 summarizes the results, where we used MLE method to estimate the weights of different acoustic models. From Table 8.21 we observe that when *m*=15, the word recognition accuracy is worse than the baseline, since now all the PDTs are very weak. As *m* increases, word recognition accuracy improves, but the accuracy differences among *m*=100, 150, 200 are small. As *m* increases further, word accuracy becomes worse, since now the correlations among the trees become too high. Among the six cases of *m*'s, it appears that *m*=150 is the best choice since it yielded highest word recognition accuracy. On the other hand, *m* in the range of 100 to 200 can all be considered as good choices. Note that RF entails a random procedure: if we run the training procedure several times, then slightly different result would be produced each time. The variations should be small when the forest size *K* is large.

Table 8.21 Average word accuracy versus the subset size *m* of phonetic questions in RF, with *K*=50.

m	Average word accuracy (%)
15	77.68
50	80.38
100	80.92
150	80.96
200	80.90
210	80.65

Model correlation versus size of question subset

We further measured correlations among acoustic models resulting from different PDT sets as a function of the question subset size *m*, since the quality of acoustic models directly impacts speech recognition accuracy. Assume that in the training dataset we have *U* phone units, *K* sets of acoustic models, and *N* triphone states. Given a speech feature vector $\underline{x}_t$, we compute the posterior probability scores of the triphone state *n* for each model *k*, i.e.,

$$P_k(n \mid \underline{x}_t) = \frac{P_k(\underline{x}_t \mid n)}{\sum_{n'=1}^{N} P_k(\underline{x}_t \mid n')} \tag{8.3}$$

where $n = 1, \cdots, N$, $t = 1, \cdots, T$, $k = 1, \cdots, K$, and the prior probabilities of triphone states are assumed uniform.

To compute correlation among the acoustic models obtained by RFs, we look at two sets of acoustic models at a time, say the *i*th and *j*th. The speech feature vectors $\underline{x}_t$ are grouped into sets Ω_u, $u = 1, \cdots, U$, based on Viterbi alignment [41]. For model sets *i* and *j* and $\underline{x}_t \in \Omega_u$, we have the posterior probability vector pairs $\underline{p}_i(t) = [P_i(1 \mid \underline{x}_t), \cdots, P_i(N \mid \underline{x}_t)]'$ and $\underline{p}_j(t) = [P_j(1 \mid \underline{x}_t), \cdots, P_j(N \mid \underline{x}_t)]'$. For each

triphone state n, we compute the correlation coefficient between the posterior probabilities $P_i(n \mid \underline{x}_t)$ and $P_j(n \mid \underline{x}_t)$ by

$$Corr_{u,n}(i,j) = \left| \frac{\sum_{t \in \Omega_u} [P_i(n \mid \underline{x}_t) - \overline{P}_i(n \mid \underline{x}_t)][P_j(n \mid \underline{x}_t) - \overline{P}_j(n \mid \underline{x}_t)]}{\sqrt{\sum_{t \in \Omega_u} [P_i(n \mid \underline{x}_t) - \overline{P}_i(n \mid \underline{x}_t)]^2 \sum_{t \in \Omega_u} [P_j(n \mid \underline{x}_t) - \overline{P}_j(n \mid \underline{x}_t)]^2}} \right| \tag{8.4}$$

The correlation between the ith and the jth acoustic model sets $Corr(i, j)$ is obtained by averaging $Corr_{u,n}(i, j)$ over the N triphone states and the U phone units. Finally, the correlation among the K sets of acoustic models is obtained by averaging over every pair (i, j).

Table 8.22 provides the correlation data measured from the dataset of Dr. 1 and the corresponding word recognition accuracies, with K set to 20 and m varied from 15 to 210. We observe that the correlation among the multiple models increases with m. Word recognition accuracy depends on the correlation among the trees as well as the strength of each tree. Here m = 150 and 200 yielded the highest word recognition accuracy, whereas using smaller (weak trees) and larger (higher correlation) m's both lowered accuracy performance.

Table 8.22 Correlation among acoustic models and word accuracy versus the phonetic questions subset size m, measured on dataset of Dr. 1 and with K=20.

m	15	50	100	150	200	210
Correlation	0.739	0.762	0.790	0.820	0.890	0.930
Word accuracy (%)	76.35	78.69	78.79	79.00	79.00	78.36

RF performance versus model complexity

We also evaluated the performance of RFs with respect to the complexity of Gaussian mixture densities, i.e., the number of Gaussian components per GMD or mixture size, with the state tying thresholds in the PDTs kept unchanged. Table 8.23 summarizes the word recognition accuracies with the number of Gaussian components

per GMD varied to be 8, 16, 20 and 24. For the RF method, we set the question subset size m = 150 and used the MLE method to estimate weights for model combination. From Table 8.23 we observe that the RF method improved accuracy performance over the baseline in every mixture size case. In baseline, using mixture size of 16 yielded best performance, but further increasing mixture size led to overfitting and thus decreased accuracy. In RF, accuracy performance improved with mixture size, and at mixture size of 24 overfitting still did not occur, indicating that the RF-based state tying is more robust to overfitting.

Table 8.23 Word recognition accuracies versus number of Gaussian component per GMD for baseline and RF methods (m=150)

	Number of Gaussian components per GMD			
	8	16	20	24
Baseline	77.65	78.96	78.68	78.15
RF method, K=10	78.08	80.47	81.57	81.70
RF method, K=20	78.06	80.81	81.86	81.92

Chapter 9

Conclusion

In this dissertation work, several important problems of large vocabulary conversational speech recognition have been investigated and new methods are proposed to improve the performance of the Telemedicine automatic captioning system. Main contributions of this work include the following five aspects:

1. A fast confusion network algorithm, which improves the speed of CN-generation from word lattice and makes it feasible to use confusion network in online ASR systems. The time complexity of CN-generation algorithm has been reduced from $O(T^3)$ to $O(T)$, which is a significant improvement.
2. A confidence annotation method for word hypothesis outputs in automatic captioning, which includes novel features and a RF based classification technique, where both have improved annotation accuracy over existing methods.
3. A new technique to improve the speed and latency performance of speech decoding engine, which utilizes complementary word confidence scores to prune uncompetitive search paths, uses SDCHMM to speed up computation of acoustic scores and local confidence scores, and incorporates the function of pre-backtrace in decoding search to reduce captioning latency.
4. Two new lookahead methods, constrained lookahead and stochastic full lookahead, for constructing improved PDTs in acoustic modeling, where the stochastic full lookahead method significantly decreases model size without sacrificing average recognition accuracy, and the decreased model size leads to increased speed of decoding search.
5. A Random Forests based PDT method, which effectively combines acoustic scores of multiple models in decoding search and achieves better accuracy and faster speed

than word hypothesis integration at recognition output, where using compacted acoustic models generated by clustering Gaussian Density Functions in RF tied states achieved decoding speed close to the baseline system while significant improving word accuracy.

Some of the research directions of this work can potentially be extended in the future to further improve online conversational large vocabulary speech recognition. One possible line of extension is to investigate additional features and classification techniques to improve the performance of confidence annotation. For example, boosting is a well-known classification method in machine learning, which can be investigated in confidence annotation. Another line of extension is to investigate other sampling methods, like bagging, in constructing RFs based PDTs. Furthermore, we may also implement a two-pass speech decoding engine, and uses simple models in the first-pass search and generates word lattices, and uses complex and more accurate models, like multiple acoustic models, or other language models to do rescoring in the second-pass search. This will potentially further improve word recognition accuracy for automatic speech recognition and the performance of Telemedicine automatic captioning.

Bibliography

[1] L. R. Rabiner, and Bing-Hwang Juang, *Fundamentals of speech recognition*, Prentice Hall Press, 1993.

[2] H. Ney, S. Ortmanns, "Dynamic programming search for continuous speech recognition," IEEE Signal Processing Magazine, vol. 16, issues 5, pp. 64-83, 1999.

[3] L. Deng, and D. O'Shaughnessy, *Speech processing — a dynamic and optimization-oriented approach*, Marcel Dekker, 2003.

[4] B. S. Atal, and M. R. Schroeder, "Predictive coding of speech signals," Proc. AFCRL/IEEE conference on speech communication and processing, pp. 360-361, 1967.

[5] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," Pattern Recognition and Artificial Intelligence, C. H. Chen, Ed. New York: Academic, pp. 374-388, 1976.

[6] H. Hermansky, "Perceptual Linear Predictive (PLP) analysis of speech," J. Acoustic Society of America, 87, pp. 1738-1752, 1990.

[7] J. G. Wilpon, C-H. Lee, and L. R. Rabiner, "Improvements in connected digit recognition using higher order spectral and energy features," Proc. ICASSP, pp. 349-352, 1991.

[8] N. Kumar, and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMS for improved speech recognition," Speech Communication, vol. 26, pp. 283-297, 1998.

[9] A. Hyvarinen, and E. Oja, "Independent component analysis: a tutorial," http://www.cis.hut.fi/aapo/papers/IJCNN99_tutorialweb/.

[10] S. J. Young, J. J. Odell and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modeling," in Proc. ARPA Human Lang. Tech. Workshop, pp. 307-312, 1994.

[11] M. Tomita, “An efficient augmented-context-free parsing algorithm,” Computer Linguistics, 13 (1-2), pp. 31-46, 1987.

[12] K. Lari, and S. Young, “Application of stochastic context-free grammars using the inside-outside algorithm,” Computer Speech and Language, 5(3), pp. 237-257, 1991.

[13] F. Jelinek, “Up from trigrams! - the struggle for improved language models,” Proc. of Eurospeech, pp. 1037-1040, 1991.

[14] S. M. Katz, “Estimation of probabilities from sparse data for the language model component of a speech recognizer,” IEEE Trans. on Acoustic, Speech and Signal Processing, vol. ASSP-35, pp. 400-401, 1987.

[15] S. Ortmanns, H. Ney, and X. Aubert, “A word graph algorithm for large vocabulary continuous speech recognition,” Computer Speech and Language, 11(1), pp. 43-72, 1997.

[16] L. Mangu, E. Brill, and A. Stolcke, “Finding consensus among words: lattice-based word error minimization,” Proc. EUROSPEECH, 1999.

[17] J. Xue and Y. Zhao, “Random forests-based confidence annotation using novel features from confusion network,” Proc. ICASSP, pp. I-1149 — I-1152, 2006.

[18] L. Mangu, E. Brill, and A. Stolcke, “Finding consensus in speech recognition: word error minization and other application of confusion network,” Computer Speech and Language, vol. 14 (4), pp. 373-400, 2000.

[19] Y. Zhao, X. Zhang, R-S. Hu, J. Xue, X. Li, L. Che, R. Hu, and L. Schopp, “An automatic captioning system for Telemedicine,” Proc. ICASSP, pp. I-957 – I-960, 2006.

[20] F. Wessel, K. Macherey, and R. Schlueter, “Using word probabilities as confidence measures,” Proc. ICASSP, pp. 225-228, 1998.

[21] G. Evermann and P.C. Woodland, “Large vocabulary decoding and confidence estimation using word posterior probabilities,” Proc. ICASSP, pp. 1655-1658, 2000.

[22] Y. Fu and L. Du, "Combination of multiple predictors to improve confidence measure based on local posterior probabilities," Proc. ICASSP, pp. 93-96, 2005.

[23] B. Dong, Q. Zhao and Y. Yan, "A fast confidence measure algorithm for continuous speech recognition," Proc. Interspeech, pp. 1457-1460, 2005.

[24] R. Zhang and A. Rudnicky, "Word level confidence annotation using combinations of features," Proc. Eurospeech, pp. 2105-2108, 2001.

[25] R. Sarikaya, Y. Gao and M. Picheny, "Word level confidence measurement using semantic features," Proc. ICASSP, pp. I-604 – I-707, 2003.

[26] T. G. Dietterich, "Ensemble methods in machine learning," Proc. of the First International Workshop on Multiple Classifier Systems, pp. 1-15, 2000.

[27] Random Forests classifier description, site of Leo Breiman.

[28] L. Breiman, "Random Forest," Machine Learning 45 (1), pp. 5-32, 2001.

[29] The R Project for statistical Computing, http://www.r-project.org.

[30] X. L. Aubert, "An overview of decoding techniques for large vocabulary continuous speech recognition," Computer Speech and Language, vol. 16, pp. 89-114, 2002.

[31] S. Ortmanns and H. Ney, "Look-ahead techniques for fast beam search," Computer Speech and Language, vol. 14, pp. 15-32, 2000.

[32] X. Li and Y. Zhao, "A fast and memory-efficient N-gram language model lookup method for large vocabulary continuous speech recognition," Computer Speech & Language, vol. 21, iss. 1, pp. 1-25, 2007.

[33] E. Bocchieri and B. K.-W. Mak, "Subspace distribution clustering Hidden Markov Model," IEEE Transactions on Speech and Audio Processing, vol. 9, no. 3, pp. 264-275, 2001.

[34] T. Kawahara, H. Nanjo, S. Furui, "Automatic transcription of spontaneous lecture speech," IEEE Workshop on ASRU, pp. 186-189, 2001.

[35] S. Abdou and M.S. Scordilis, "Beam search pruning in speech recognition using a posterior probability-based confidence measure," Speech Communication, vol. 42,

pp. 409-428, 2004.

[36] X-B. Li, F. K. Soong, T. A. Myrvoll and R-H. Wang, “Optimal clustering and non-uniform allocation of Gaussian kernels in scalar dimension for HMM compression,” Proc. ICASSP, pp. I669-I772, 2005.

[37] D. Povey and P.C. Woodland, “Frame discrimination training of HMMs for large vocabulary speech recognition,” Proc. ICASSP, pp. 333-336, 1999.

[38] K. Hirose, A. Sakurai and H. Honno, “Use of prosodic features in the recognition of continuous speech,” Proc. ICSLP, pp. 1123-1126, 1994.

[39] C. Chesta, P. Laface and F. Ravera, “Bottom-Up and Top-Down State Clustering for Robust Acoustic Modeling,” Proc. EUROSPEECH, pp. 11-14, 1997.

[40] S. Esmeir and S. Markovitch, “Lookahead-based Algorithms for Anytime Induction of Decision Trees,” Proc. ICML, vol. 69, pp. 257-264, 2004.

[41] HTK Toolkit, http://htk.eng.cam.ac. uk

[42] A. Lazaridès, Y. Normandin and R. Kuhn, “Improving Decision Tree for Acoustic Modeling,” Proc. ICSLP, pp. 1053-1056, 1996.

[43] S. Wang and Y. Zhao, “Online Bayesian tree-structured transformation of HMMs with optimal model selection for speaker adaptation,” IEEE Trans. Speech Audio Processing, vol. 9, no. 6, pp. 663-677, 2001.

[44] K. Shinoda and C.-H. Lee, “A structural Bayes approach to speaker adaptation,” IEEE Trans. Speech Audio Processing, vol. 9, no. 3, pp. 276-287, 2001.

[45] W. Reichl and W. Chou, “Robust decision tree state tying for continuous speech recognition,” IEEE Trans. Speech Audio Processing, vol. 8, no. 5, pp. 555-566, 2000.

[46] I. Shafran and M. Ostendorf, “Acoustic model clustering based on syllable structure,” Computer Speech Language, vol. 17, no. 4, pp. 311-328, 2003.

[47] S. S. Chen, and P. S. Gopalakrishnan, “Clustering via the Bayesian information criterion with applications in speech recognition,” Proc. ICASSP, pp. 645-648,

1997.

[48] J-T. Chien and S. Furui, “Predictive hidden Markov model selection for speech recognition,” IEEE Trans. Speech Audio Processing, vol. 13, no. 3, pp. 377-387, 2005.

[49] R. Hu and Y. Zhao, “Bayesian decision tree state tying for conversational speech recognition,” Proc. ICSLP, pp. I738-I741, 2006.

[50] J. Xue and Y. Zhao, “Novel lookahead decision tree state tying for acoustic modeling,” Proc. ICASSP, pp. IV-1133 — IV-1136, 2007.

[51] D. Madigan, A. E. Raftery, C. Volinsky and J. Hoeting, “Bayesian model averaging,” AAAI Workshop on Integrating Multiple Learned Models, pp. 77-83, 1996.

[52] D. Aha and R. Bankert, “Cloud classification using error-correcting output codes,” Artificial Intelligence Applications: Natural Resources, Agriculture and Environmental Science, vol. 11, pp. 13-28, 1997.

[53] L. Breiman, “Bagging predictors,” Machine Learning, vol. 24, no. 2, pp. 123-140, 1996.

[54] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” Journal of Computer and System Sciences, vol. 55, issue 1-3, pp. 119-139, 1997.

[55] J. H. Friedman, “Stochastic gradient boosting,” Computational Statistics & Data Analysis, vol. 38, issue 4, pp. 367-378, 2002.

[56] A. Demiriz, K. P. Bennett and J. Shawe-Taylor, “Linear programming boosting via column generation,” Machine Learning, vol. 46, pp. 225-254.

[57] O. Siohan, B. Ramabhadran, and B. Kingsbury, “Constructing ensembles of ASR systems using randomized decision trees,” Proc. ICASSP, pp. I-197 – I-200, 2005.

[58] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” J. Royal Stat. B, vol. 39, no. 1, 1-38, 1977.

[59] R. Hu, J. Xue, and Y. Zhao, “Incremental largest margin linear regression and MAP adaptation for speech separation in Telehealth application,” Proc. Eurospeech, pp. 261-264, 2005.

[60] R. Kneser, and H. Ney, “Improved backing-off for m-gram language modeling,” Proc. ICASSP, pp. 181-184, 1995.

[61] A. Stolcke, “SRILM – An extensible language modeling toolkit,” Proc. ICSLP, pp. 901-904, 2002.

[62] X. Zhang, Y. Zhao, and L. Schopp, “A novel method of language modeling for automatic captioning in TC video teleconferencing,” IEEE Trans. Information Technology in Biomedicine, vol. 11, pp. 332-337, 2007.

[63] J. G. Fiscus, “A post-processing system to yield reduced word error rate: Recognizer output voting error reduction (ROVER),” Proc. IEEE ASRU Workshop, pp. 347-352, 1997.

Vita

Jian Xue was born on Sep. 3, 1975, in Wuhu, Anhui, China. He received B.E. and M.S. degrees in Electronic Engineering, both from Southeast University at Nanjing China. He expects to receive Ph.d degree in Computer Science from University of Missouri – Columbia at Columbia, Missouri, US, by the end of 2007.

From 2000 to 2003, he was a R&D engineer in ZTE Corporation, China. He is currently a speech scientist in IBM T.J. Watson Research Center at Yorktown Heights, NY, US. His research interests include automatic speech recognition, machine translation, machine learning, and digital signal processing.